НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи та методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Система автоматизації адміністрування сайту»

Виконав :
Студент IV курсу, групи КА-66
Жуковський Андрій Віталійович _____

Керівник:
асистент кафедри ММСА Древаль Максим Михайлович _____

Консультант з економічного розділу:
доцент Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є. _____

Рецензент:
Прізвище, ім'я, по батькові _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інтелектуальний аналіз даних в управлінні проектами»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«25» травня 2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Жуковському Андрію Віталійовичу

1. Тема роботи «Система автоматизації адміністрування сайту», керівник роботи Древаль Максим Михайлович, асистент кафедри ММСА, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентом роботи 8.06.2020.

3. Вихідні дані до роботи: існуюче програмне забезпечення для автоматизації адміністурвання; перелік необіходного функціоналу, необхідного для реалізації системи автоматизації адміністрування.

4. Зміст роботи: методи автоматизації адміністрування; деталізація технічних засобів реалізації; результати роботи розроблювальної програми; функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу: ключові поняття автоматизації, та адміністрування; архітектурний стиль REST, шари веб-сервісу, схема роботи модулів Spring.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	13.04.2020	
2	Підготовка першого розділу	22.04.2020	
3	Підготовка другого розділу	1.05.2020	
4	Розробка програмного продукту	15.05.2020	
5	Підготовка третього розділу	18.05.2020	
6	Підготовка економічної частини	30.05.2020	
7	Оформлення розділів відповідно до нормоконтролю	2.06.2020	
8	Оформлення дипломної роботи	4.06.2020	
9	Підготовка презентації доповіді	7.06.2020	

Студент Андрій Жуковський

Керівник Максим Древаль

РЕФЕРАТ

Дипломна робота: 118 с., 40 рис., 10 табл., 3 додатки, 26 джерел.

SPRING FRAMEWORK, АДМІНІСТРУВАННЯ, АВТОМАТИЗАЦІЯ, API, REST, SPRING SECURITY, VUEJS

Метою роботи є розробка системи автоматизації адміністрування сайту для надання спортивних послуг на ринку Укаїни.

В роботі проведено дослідження та аналіз ринку надання спортивних послуг, методи і методології для автоматизації адміністрування та виявлення функціона необхідних для вирішення проблематик адміністрування.

У ході дослідження було встановлено необхідні функції для зменшення затрат часу та ресурсів на проведення адміністративних операції, як зі сторони адміністрації та зі сторони користувачів.

Було реалізовано веб-сервіс для управління, обробки та внесення даних з необхідними функціями для автоматизації адміністрування, також було реалізовано веб-інтерфейс, який взаємодіє з веб-сервісом. Планується розвивати роботу у напрямку розростання функціоналу, також перехід на нативні додатки різних платформ.

.

ABSTRACT

Diploma Thesis (Bachelor's Thesis): 118 p., 40 fig., 10 tabl., 3 annexes, 26 sources.

SPRING FRAMEWORK, ADMINISTRATION, AUTOMATION, API, REST, SPRING SECURITY, VUEJS

The purpose of the work is to develop a system of automation of site administration for the provision of sports services in the Ukrainian market.

The research and analysis of the market of sports services, methods and methodologies for automation of administration and identification of functions necessary for solving administrative problems are carried out in the work.

The study identified the necessary functions to reduce the time and resources spent on administrative operations, both on the part of the administration and on the part of users.

A web service for managing, processing and entering data with the necessary functions for automation of administration was implemented, as well as a web interface that interacts with the web service. It is planned to develop work in the direction of growing functionality, as well as the transition to native applications of different platforms.

ЗМІСТ

ВСТУП

Потреба в автоматизації виникла ще в стародавні часи і неухильно зростала в процесі розвитку людських потреб. Винахід будь-яких нових засобів пересування чи виробництва тоді вимагав удосконалень, підвищення комфорту, безпеки для інших та, звичайно, простоти в експлуатації. Автоматизація у всі часи людства розвивала людину, давала змогу здобувати бажаного з меншими зусиллями й затратами. Завжди були й будуть задачі, процеси, які можливо автоматизувати, без цього наш світ не може змінюватися.

Поява й розповсюдження персональних комп'ютерів привели до діджиталізації у всіх сферах життя людини. Це дало неабиякий приріст швидкості розвитку людства. В наш час залишилось мало спеціальностей, які не використовують в своїх задачах комп'ютер. Теперішній людини навіть в повсякденному житті важко представити себе без комп'ютера, смартфона і так далі. Саме це створило нову еру автоматизації.

Невід'ємною частиною стала автоматизація у компаніях. Це сприяє зменшенню затрат та збільшення якості. Робота адміністратора зав'язана на великій кількості даних і інформації, тому для адміністраторів, для людей які працюють з даними, створюють велику кількість програмного забезпечення, яке допомагає і автоматизує роботу.

Сфера надання спортивних послуг мало розвинута в цьому напрямку, тому було обрано тему, яка допоможе розвинути автоматизацію в цій сфера, а в подальшому сприяє її розвитку.

Задачі роботи:

1) Провести дослідження ринку надання спортивних послуг, та в суміжних сферах.
2) Провести дослідження задля знаходження необхідного функціоналу, який сприяє збільшенню продуктивності, та зменшенню затрат.
3) Проаналізувати отримані результати, визначити та описати основні переваги, недоліки та особливості.

Згідно до головних задач даної роботи для проведення дослідження є доречним розробити програмний продукт, що відповідає наступним вимогам:

1) Має працювати на різних апаратних платформах, бути легко доповнюваним, мати інтуїтивно зрозумілий інтерфейс.

2) Зробити веб-сервіс, який буде надавати користувачеві, тренеру і адміністратору змогу автоматизувати процеси необхідні в наданні та придбанні спортивних послуг.

3) Надавати адміністратору змогу вилучати, долучати, змінювати необхідні дані.

4) Надавати користувачу змогу купівлі абонементів, запису на тренування, налаштування свого профілю, перегляду розкладу, змогу завантажувати свої фото, відео.

5) Надавати тренеру можливість вести облік студентів, формування й перегляд свого розкладу, вести свої сторінку, бачити свою статистику.

РОЗДІЛ 1 МЕТОДИ АВТОМАТИЗАЦІЇ АДМІНІСТРУВАННЯ

1.1 Розвиток адміністрування і менеджменту в спорті

У сучасній науці під "управлінням" розуміється процес керівництва або управління працівником, робочою групою, колективом або різними організаціями, які працюють в умовах ринкової економіки. Управління спортом - це самостійний вид професійної діяльності, який спрямований на досягнення цілей спортивної та спортивної організації, яка працює в ринковому середовищі за рахунок раціонального використання матеріальних, трудових та інформаційних ресурсів. Іншими словами, спортивний менеджмент - це теорія та практика ефективного управління організаціями спортивної галузі в умовах ринку.

Управління є одним з найважливіших факторів функціонування та розвитку фізичної культури та спорту Історично в нашій країні та країнах СНГ траплялося так, що тренери, інструктора та методисти брали участь у спортивному управлінні. Вони часто поєднували навчально-виховну діяльність з клубами, спортивними товариствами, спортивними федераціями, хоча їх описи роботи не передбачали виконання таких обов'язків.

У вісімдесятих роках в Європі стали з'являтися навчальні заклади, що готують спортивних менеджерів за програмами MBA або за національними освітніми стандартами. Цей період збігся за часом з початком комерціалізації футболу, перетворення його в прибутковий бізнес. Лідером в цьому процесі є Великобританія, де працює єдина в світі спеціалізована MBA-програма в Університеті Ліверпуля (її засновник - британський футбольний гуру Роган Тейлор). Серед її випускників такі відомі персони в світі спортивного бізнесу, як керівник відділу маркетингу Англійської футбольної асоціації (АФА) Джеймс Уоррел, директор по футболу телекомпанії «Скай ТВ» Роберт Ельстоун, професор Белградській академії спорту Мілан Томич і ін.

Важлива відмінність MBA від аналогічних навчальних закладів, скажімо, в США і Канаді, спеціалізованих насамперед на менеджменті в бейсболі, боксі і гольфі - універсальність: відучившись в Швейцарії або в Великобританії, ви з

успіхом зможете взяти на себе «госпчастина» вітчизняної футбольної команди. І найголовніше, ці навчальні заклади готують саме менеджерів, а не тренерів або дієтологів.

1.1.1 Характеристика і принципи менеджменту й адміністрування в спорті

Менеджмент – управління, сукупність принципів, методів, інструментів та видів підвищення продуктивності та доходу. Тому основна мета управління - забезпечити рентабельність підприємства за допомогою продуманої організації виробничого процесу.

За останні десять років умови праці у сферах фізичної культури та спорту, відпочинку та туризму кардинально змінилися - з’явилися нові форми діяльності, нові бази, обладнання, інвентар, технології. Туристичні організації, які хочуть вижити та процвітати у спорті, особливо у складних змагальних умовах, приділяють особливу увагу навчанню працівників та підготовці фахівців зі спортивного менеджменту.

Управління спортом - це молода, але швидко зростаюча і нерівна сфера наукових і практичних знань. Згідно з опитуваннями, сучасні керівники спортивного менеджменту - організатори спорту: керівники спортивних організацій, клубів і клубів, керівники спортивних команд, тренери національних команд. Керівники спортивних споруд, директори спортивних споруд тощо. Підготовка висококваліфікованих спортивних менеджерів відіграє дуже важливу роль у цьому контексті. Ефективне функціонування різних фізичних культур та спортивних організацій вимагає збільшення кількості молодих фахівців, які володіють знаннями управлінської культури.

Основне, що забезпечує менеджмент - це не керування речами, а управління організацією та технікою роботи людей за принципами та маркетинговими програмами.

Зарубіжні експерти з управління спортивним відомством вважають, що це чотири основні функції: планування, організація, командування, аналіз.

1.1.2 Приклади програмного забезпечення для адміністрування, аналіз

Fedena - це програмне забезпечення з відкритим кодом для адміністрування шкіл, яке в основному фокусується на обробці записів.

Плюси - Fedena пропонує необмежену кількість адміністративних та студентських входів для використання їх системи, а також необмежену кількість курсів та пакетів. Система була розроблена за допомогою Ruby on Rails, тому школи можуть легко налаштувати код під потреби школи. Система включає людські ресурси, календар, управління фінансами, управління іспитом та вхід студентів / батьків.

Мінуси - Fedena є безкоштовна версія іншого однойменного програмного забезпечення для адміністрації школи. Порівнюючи безкоштовну версію з платною версією, стає зрозуміло, що у відкритій версії не вистачає низки функцій, включаючи інвентар, спеціальні звіти, реєстрацію та дисципліну.

Особливості: управління курсом, прийом, факультет обміну повідомленнями, управління шкільним календарем, управління іспитом, відстеження відвідуваності, інформаційні панелі, інформація для студентів, управління працівниками, викладачами, людські ресурси. На рис. 1.1 представлений інтерфейс програмного продукту Fedena.

Рисунок 1.1 Приклад роботи Fedena

Monday Business Management Software

Дає вам інформацію про витрати бюджету. Ви отримаєте чіткий огляд статусу проекту. Він може забезпечити зберігання файлів від 5 Гб до необмеженого.

Надає такі функції, як розширений пошук, налаштування форм та відстеження часу.

Вердикт - це програмне забезпечення для управління бізнесом має функції для планування проектів, командних завдань та часової шкали проекту. Це допоможе вам автоматизувати повторювані завдання. Він забезпечує безпеку через двофакторну автентифікацію, аутентифікацію Google, журнал аудиту, управління сеансами тощо. На рис. 1.2 представлена робота Monday Business Management.

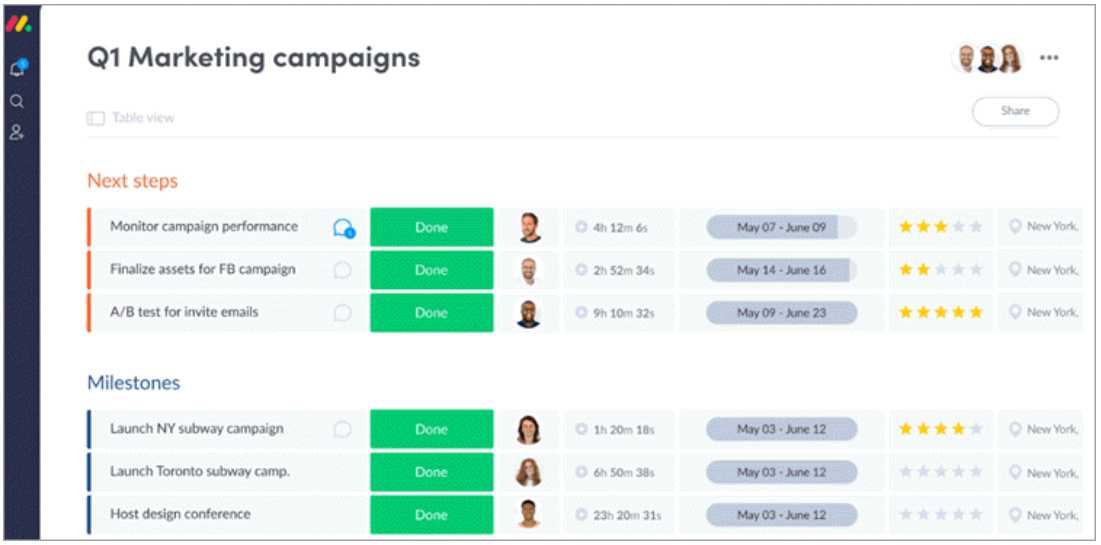


Рисунок 1.2 Приклад роботи Monday Business Management

Sage

Sage - це хмарне програмне забезпечення для управління бізнесом та набір, який надасть вам нагляд у багатьох сферах, таких як фінанси та кадрові ресурси. Він надаватиме інформацію в реальному часі про HR, Фінанси та інші щоденні операції підприємств, що допоможе вам у прийнятті обґрунтованих рішень.

Особливості - для управління бізнесом Sage пропонує цілий ряд продуктів, які включають управління підприємством, основні засоби, 100cloud, CRM, звітність, будівництво та нерухомість, радник інвентаризації тощо.

Він має функції взаємовідносин із клієнтами, обробки платежів, управління послугами, продажу та електронної комерції, людських ресурсів, фінансів, виготовлення, управління документами, бізнес-аналітики тощо.

Вердикт - управління хмаровим бізнесом Sage - це повний набір інтегрованих програм для фінансів, продажів, обслуговування клієнтів тощо. На рис. 1.3 зображений приклад роботи Sage.

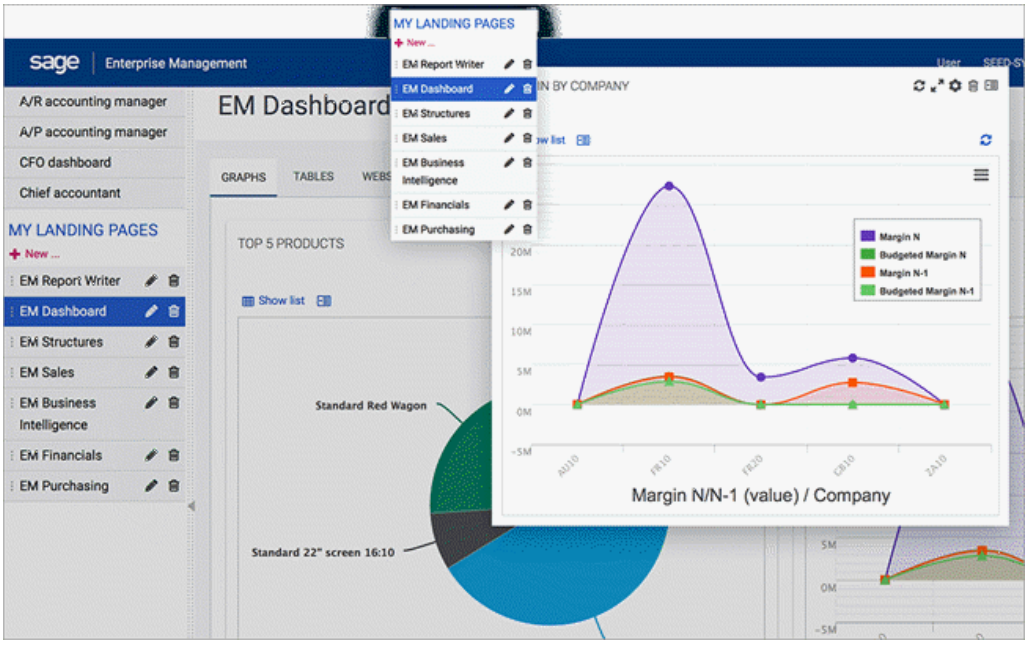


Рисунок 1.3 Приклад роботи Sage

StudioCloud

StudioCloud пропонує рішення «все в одному», яке допоможе вам в управлінні бізнесом. Він має функції для управління клієнтами, клієнтами, організаціями, партнерами та постачальниками. Це допоможе вам у плануванні та виставлення рахунків. Це допоможе вам в управлінні працівниками та поколінням.

Особливості - Для управління проектами він має функції для створення трубопроводів, термінів, завдань для проектів тощо. Він надає можливість імпорту та експорту даних.

Це дозволить вам налаштувати інтерфейс програмного забезпечення, рахунків-фактур та контрактів.

Він забезпечує функціональність для відстеження часу працівників. Він має функції прийому кредитних карток та електронних підписів.

Вердикт - StudioCloud має функції для бухгалтерії, управління проектами, маркетингових кампаній та онлайн бронювання. Він може бути інтегрований з різними сторонніми продуктами. Він може бути інтегрований із швидкими книгами, MailChimp та Google Календарями. На рис. 1.4 зображено прилкад роботи StudioCloud.

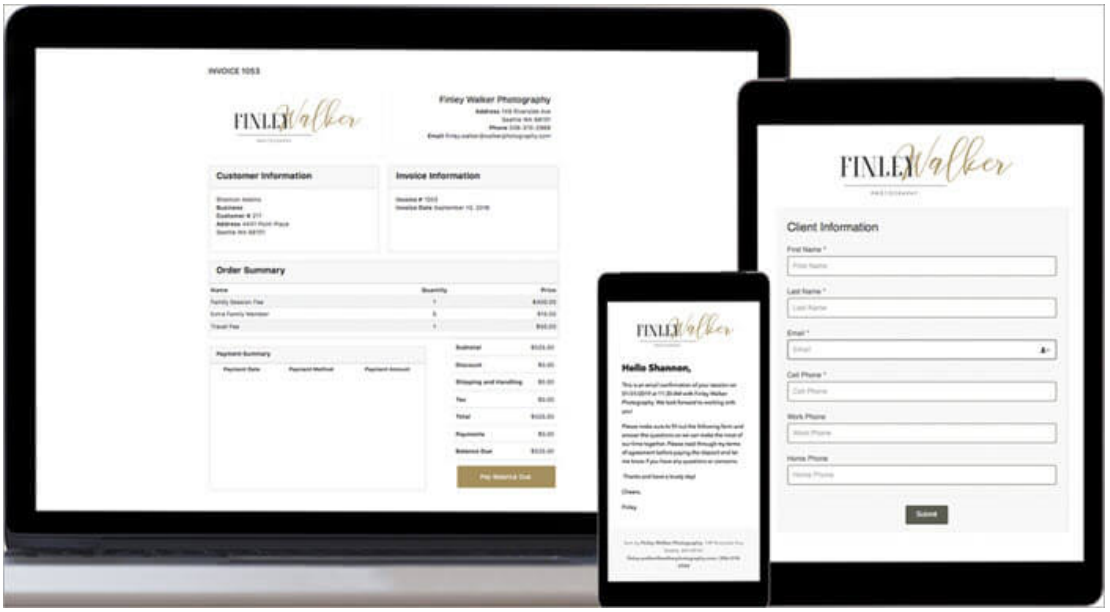


Рисунок 1.4 Приклад роботи StudioCloud

Qualsys

Для пакету реалізації можна вибрати інтеграцію ERP або API, спеціальну розробку, додаткове навчання, спеціальні шаблони або підтримку перевірки.

Qualsys пропонує десять програмних модулів для вашого інтегрованого програмного забезпечення для управління бізнесом. Компанія дозволить вам використовувати будь-яку комбінацію модулів. Це буде єдиним рішенням для всіх ваших даних та діяльності.

Особливості - Qualsys має програмне забезпечення для управління документами, програмне забезпечення для обслуговування обладнання, програмне забезпечення для аварій та інцидентів, програмне забезпечення для

управління ризиками, програмне забезпечення для управління постачальниками, замовляє модулі, програмне забезпечення для управління документацією, скаргами, програмне забезпечення для управління скаргами, програмне забезпечення управління аудитом та програмне забезпечення CAPA.

Він забезпечує інтегровану систему управління бізнесом. Він пропонує різні модулі та системи управління.

Як рішення для управління бізнесом, Qualsys має функції управління документами, політикою та SOP, повна видимість постачальників, звітність про бізнес-аналітику, управління компетентністю тощо.

Вердикт - Qualsys є повним набором програмного забезпечення для управління бізнесом, який пропонує рішення щодо ризиків, документів, аудитів, політики тощо. На рис. 1.5 зображено інтерфейс Qualsys.

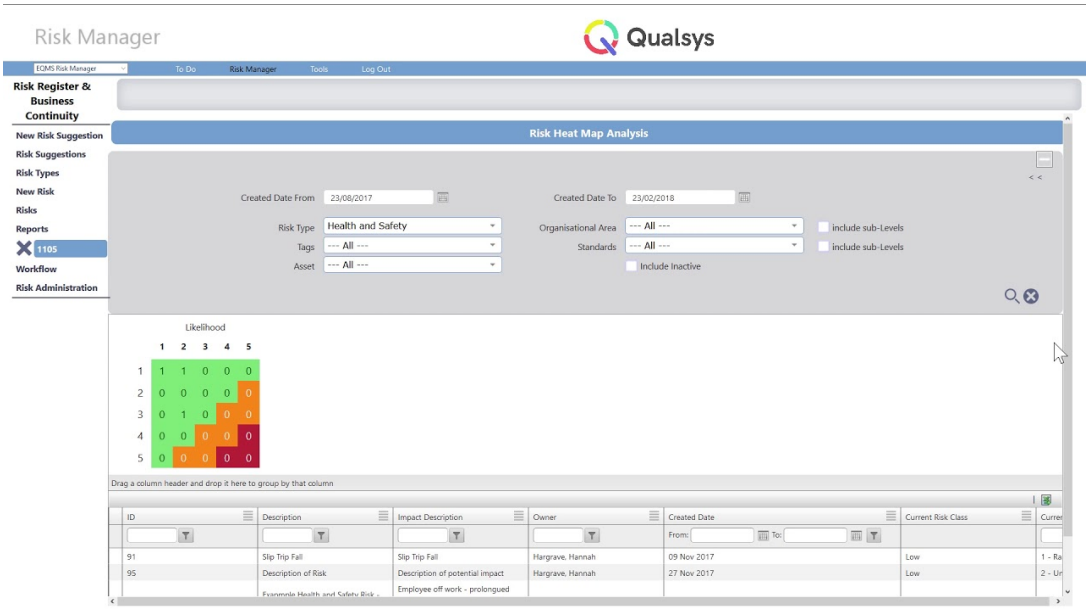


Рисунок 1.5 Приклад роботи Qualsys

1.1.3 Методи вирішення проблем адміністрування

Основною роботою адміністратора являється зазвичай робота з даними, тому більшість задач адміністрування вирішується за допомогою комп’ютера.

Існує багато програмних рішень, вони можуть бути локальними та хмарними, безкоштовними та платними, вузькоспрямованими та загальними.

Всі ці рішення мають обширний функціонал і є досить великоваговими. В користуванні деякі з них мають досить складний інтерфейс і не завжди неквалі-фікований користувач в силі розібратися з ним.

1.1.4 Аналіз та вибір методів для вирішення проблематики адміністрування

Процеси глобалізації найбезпосереднішим чином впливають на розвиток спортивних технологій. У міру розвитку суспільства фізична активність і спорт все ширше проникають в усі сфери життя людей, стають все більш значущою і невід'ємною частиною світової цивілізації. Бурхливими темпами розвивається в останні десятиліття і спорт вищих досягнень, поступово перетворюючись в окрему сферу діяльності. Спортивна наука все більше перетворюється в самостійну наукову дисципліну, до якої залучаються фахівці з різних спеціальностей. Для моніторингу та аналізу дій спортсмена використовуються найостанніші досягнення наукової думки - від мікроелектроніки до молекулярної біології.

Безпосереднім результатом наукового прогресу є зміни спортивної техніки і досягнення вищих результатів, які ще вчора здавалися немислимими. Підвищення ефективності тренувального процесу на кожному етапі тренувального процесу може бути здійснено тільки в результаті об'єднання фрагментарних знань, отриманих тренерами, спортивними фахівцями і науковцями.

Труднощі створення концепції індивідуальної тренування на даний момент полягає у відсутності чіткої інтеграційної моделі, узагальнюючої розрізнені досягнення в різних сферах наукової діяльності. Спортсменам і тренерам доводиться зараз працювати в ситуації постійних нововведень. Інновації, які може використовувати тренер, різноманітні: нові методики спортивного тренування, ділові ігри, проблемне навчання, діалогове викладання тощо. Підвищення інтелектуального рівня тренерів, методистів і всіх фахівців,

що працюють в спорті вищих досягнень, є першочерговим завданням всіх провідних спортивних держав .

Оскільки сфера спорту завжди була більше практична, ніж технічна та теоретична, нема чітких методів та методологій для вирішення проблем автоматизації адміністрування в сфері спорту. Спеціалісти, які працюють в цій сфера, не завжди готові оволодіти складними технічними засобами, які вже є на ринку, тому більшість товариств, організацій, компаній, клубів досі використовують папір для зберігання інформації, звітності тощо. Більші компанії використовують комп'ютер щоб ввести звітність, для адміністрування. Зазвичай користуються Excel в рідких випадках більш складними програмами.

В той час як більшість компаній переносять всі свої функції в хмарний простів інтернету, сфера спорту робить тільки маленькі кроки до цього.

По скільки в спорті є завжди двосторонній, а то і більше зв'язки між людьми. Має бути зв'язок між тренером, адміністратором, спортсменом. В сфері надання спортивних послуг треба щоб користувач, спортсмен, викладач, тренер, адміністратор могли доступним чином отримувати інформацію, замінювати, видаляти, та взаємодіяти між собою, великі компанії почали використовувати веб-сервіси, сайти тощо для взаємодії. Тому що це є доступним для всіх, і в нас час кожна людина здатна оволодіти пошуком, зміною інформації на сайті.

Під час спалаху коронавірусу сфера спорту отримали величезні збитки через те, що вона використовую пряму зв'язку між людиною та людиною, але це стало поштовхом для розвитку спорту в іншому напрямку.

Розглядаючи різні компанії в сфері спорту, можна виділити лідерів в плані діджиталізації та перенесення деяких адміністративних задач в більш автоматизований режим.

Розглянемо компанію в танцювальній сфері Myway.

Це не сумнівно один з лідерів в наданні спортивних послуг в напрямку танцювальної індустрії.

Основною відмінністю від інших компаній являється доступний сайт, з великою кількістю доступної інформації. https://mywaydance.com . На рис. 1.6 та

1.7 ми бачимо головну сторінку на сайті myway, інформацію яку можливо отримати на ньому.

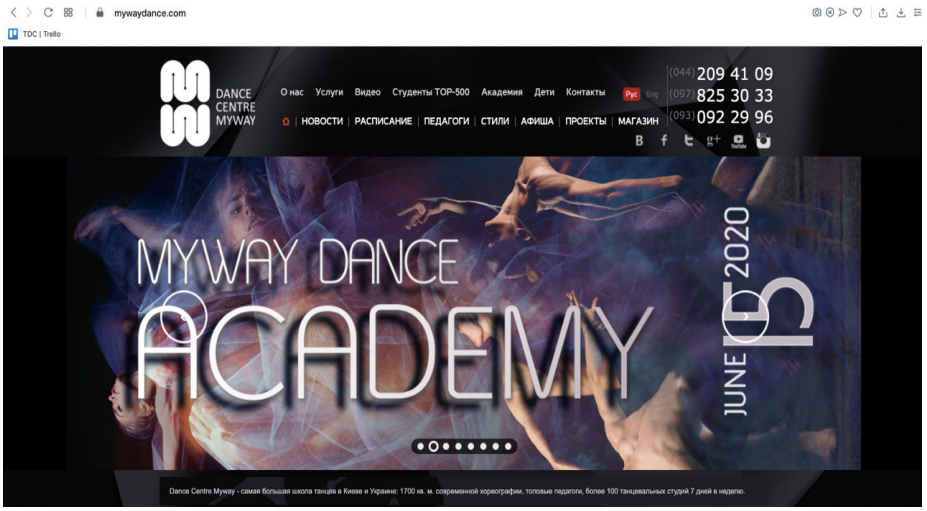


Рисунок 1.6 Приклад сайту Myway

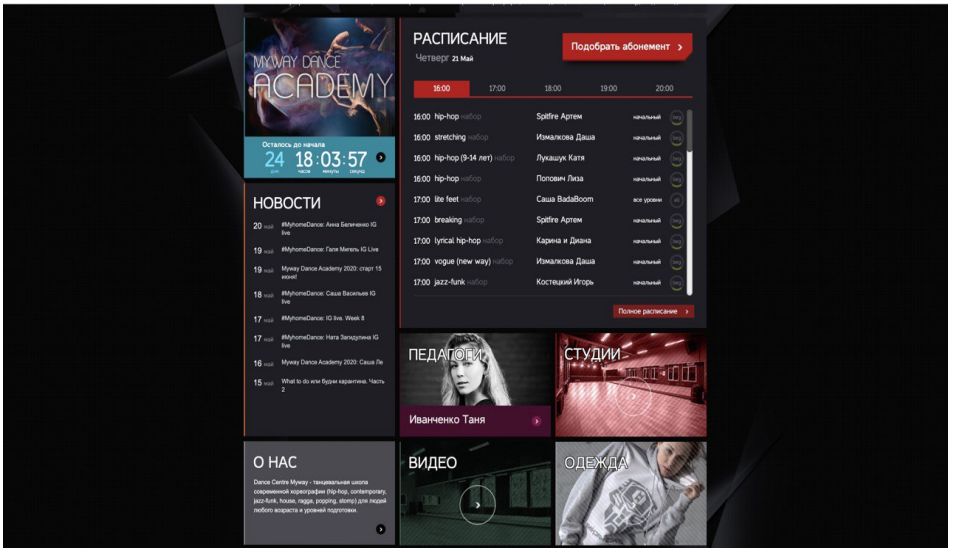


Рисунок 1.7 Приклад сайту Myway

На рис. 1.8 та 1.9 ми бачимо функціонал реалізований на сайті myway.

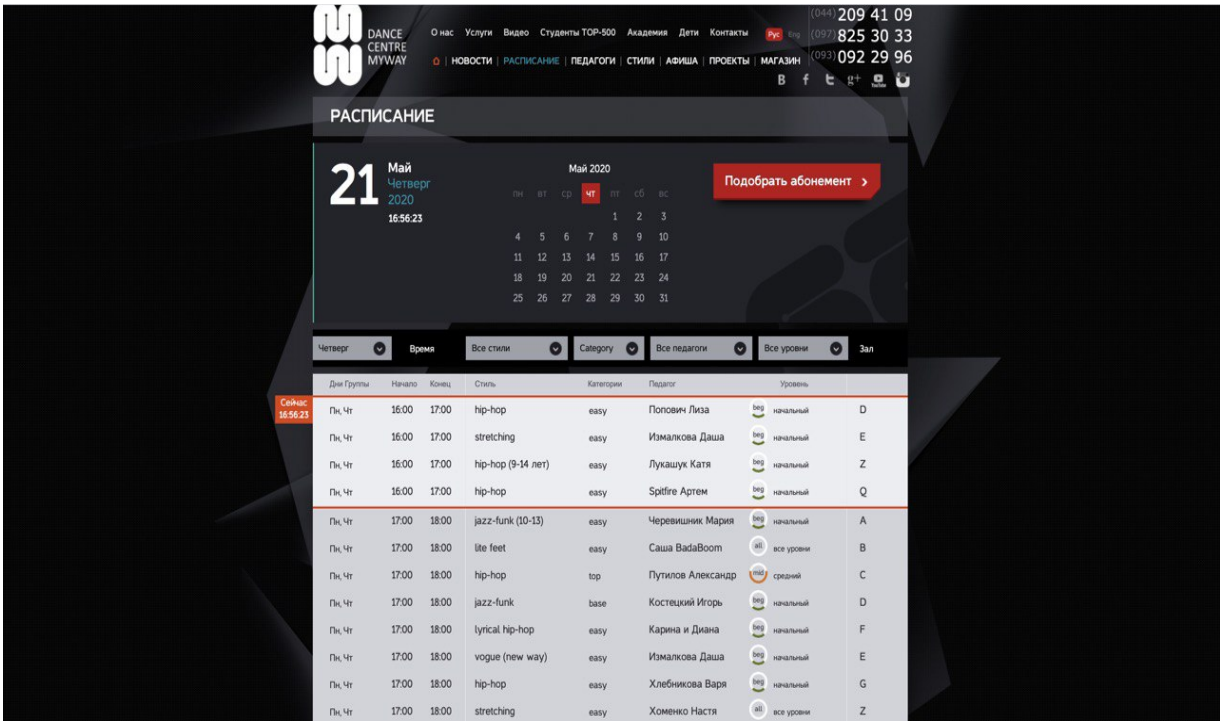


Рисунок 1.8 Приклад відображення розкладу на сайті myway

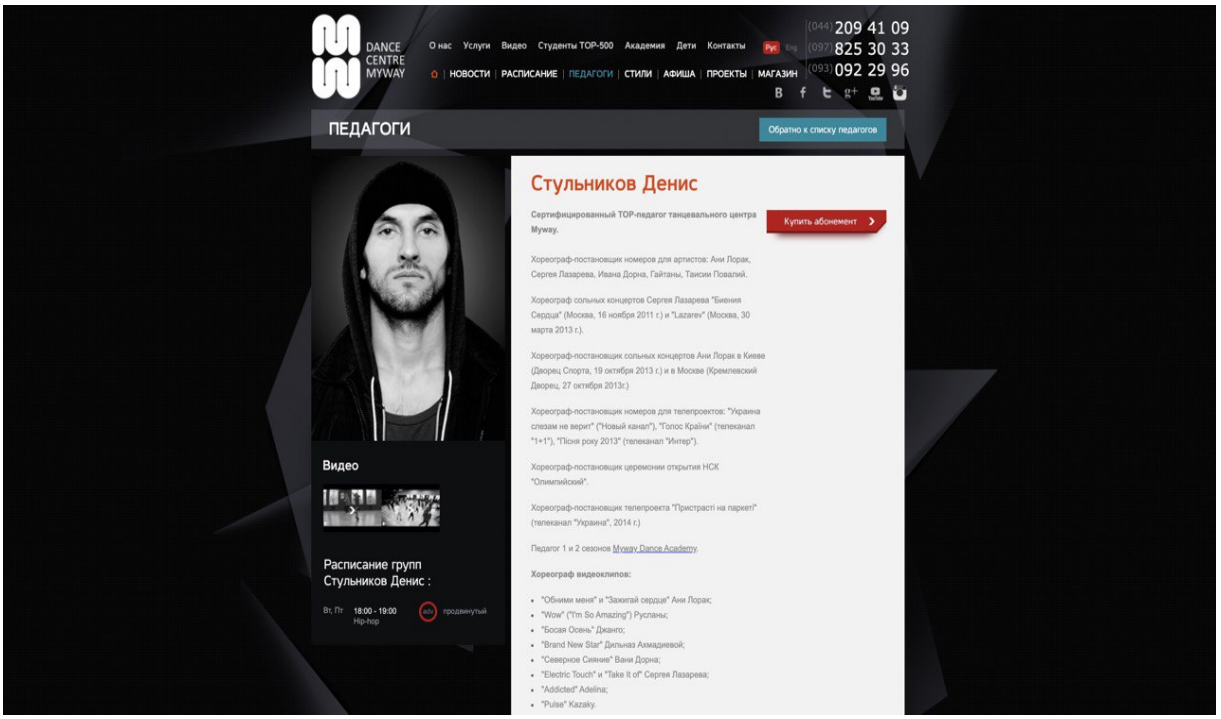


Рисунок 1.9 Приклад відображення персональної сторінки тренера myway

Ми бачимо що на сайті myway, хоча це велика організація, з великою кількістю спортсменів, інформація на сайті відносно статична і може змінюватися виключно адміністратором.

Із функціоналу приваблює досить велика кількість інформації про тренерів, студентів, можливість перегляду розкладу, та купівлі абонементу.

FlyMark - Досить велика платформа появилась в європейскому просторі, ця платформа згодом надійшла до України, й містить в собі базу всіх танцівників та тренерів спортивно-бального танцю. На рис. 1.10 продемонстровано головну сторінку сайту FlyMark.

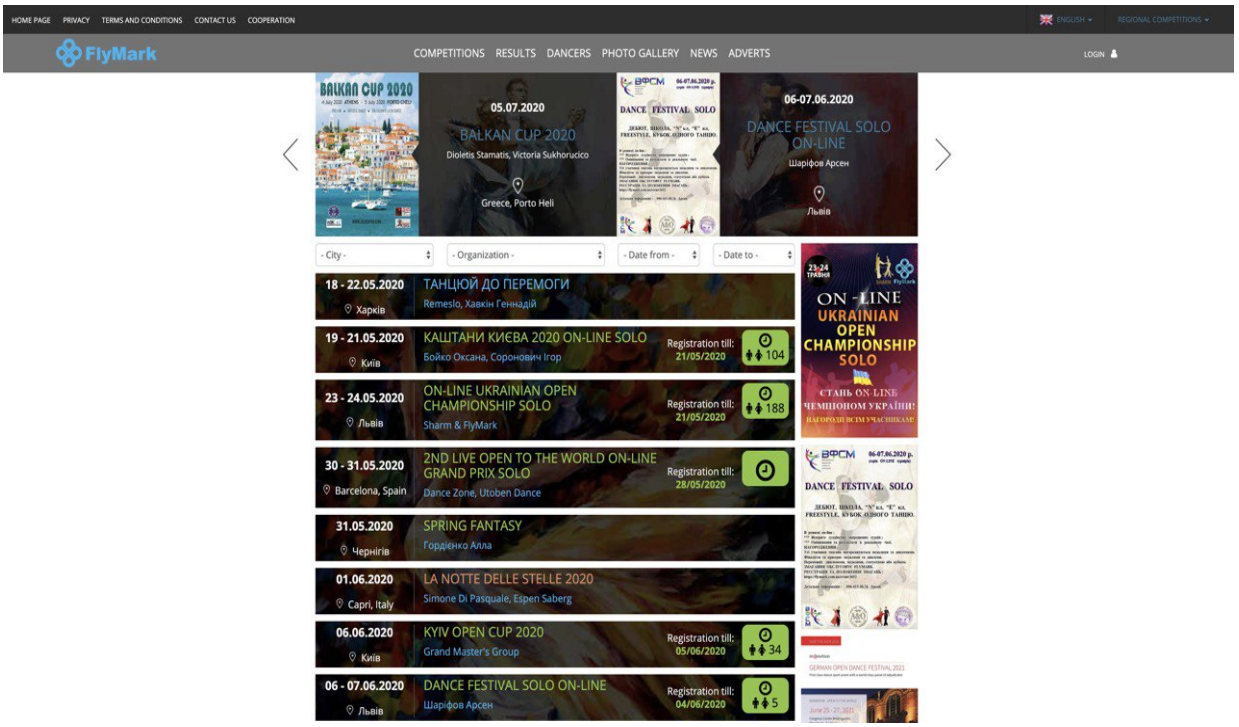


Рисунок 1.10 Приклад сайту FlyMark

Через цю платформу проходять більше ніж 80% турнірів та змагань по всьому світу, вони містить в собі онлайн результати турніру, проводить трансляції, містить всю інформацію про статус спортсмена, клас, рейтинг тощо.

Оскільки диплом і реалізацію продукту, було зроблено задля нової організації спорту, треба зробити продукт, яких з легкістю зможе розповсюдити інформацію, вона має бути доступна, користувачі, спортсмени, тренери та адміністрація має перенести туди всі адміністративні відносини, а також зв’язаними з цим практичними задачами, мною було вирішено розробляти адаптивну систему для автоматизації адміністрування та менеджменту як зі сторони адміністратора, так і зі сторони тренера та спортсмена.

РОЗДІЛ 2 ДЕТАЛІЗАЦІЯ ТЕХГІЧНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір типу архітектури

Архітектурний стиль представляє собою набір обмежувальних умов, які застосовуються при будь-якому конструюванні. Архітектурний стиль програмних засобів є описом компонентів, які можуть бути використані для реалізації програмної системи.

Для вибраного мною підходу вирішення проблематики дипломною роботи було прийнято рішення використовувати Web API.

API - application programming interface (інтерфейс програмування додатків), набір правил і механізмів, за допомогою яких один додаток або компонент взаємодіє з іншими

Простота використання і підтримки. Хороший API просто використовувати і підтримувати.

Гарна конверсія в середовищі розробників. Якщо всім подобається ваш API, до вас приходять нові клієнти і користувачі.

Вище популярність вашого сервісу. Чим більше користувачів API, тим вище популярність вашого сервісу.

Краще ізоляція компонентів. Чим краще структура API, тим краще ізоляція компонентів.

Гарне враження про продукт. API - це як би UI розробників; це те, на що розробники звертають увагу в першу чергу при зустрічі з продуктом. Якщо API кривої, ви як технічний експерт не будете рекомендувати компаніям використовувати такий продукт, набуваючи щось стороннє.

До Web API відносяться XML-RPC і JSON-RPC, SOAP і REST.

RPC (remote procedure call - «віддалений виклик процедур») - поняття дуже старе, що поєднують давні, середні і сучасні протоколи, які дозволяють викликати метод в іншому додатку. XML-RPC - протокол, що з'явився в 1998 р незабаром після появи XML. Спочатку він підтримувався Microsoft, але незабаром Microsoft повністю переключилася на SOAP, тому в .Net Framework

ми не знайдемо класів для підтримки цього протоколу. Незважаючи на це, XML-RPC продовжує жити до сих пір в різних мовах (особливо в PHP) - мабуть, заслужив любов розробників простотою.

SOAP також з'явився в 1998 р стараннями Microsoft. Він був анонсований як революція в світі ПО. Не можна сказати, що все пішло за планом Microsoft: була величезна кількість критики через складність і ваговитості протоколу. У той же час, були і ті, хто вважав SOAP справжнім проривом. Протокол продовжував розвиватися і плодитися десятками нових і нових специфікацій, поки в 2003 р W3C не затвердила в якості рекомендації SOAP 1.2, який і зараз - останній. Сімейство у SOAP вийшло значне: WS-Addressing, WS-Enumeration, WS-Eventing, WS-Transfer, WS-Trust, WS-Federation, Web Single Sign-On.

Потім, що закономірно, все ж з'явився дійсно простий підхід - REST. Абревіатура REST розшифровується як representational state transfer - «передача стану уявлення» або, краще сказати, представлення даних в зручному для клієнта форматі. Термін "REST" був введений Роєм Філдінгом в 2000 р Основна ідея REST в тому, що кожне звернення до сервісу переводить клієнтську програму в новий стан. По суті, REST - не протокол і не стандарт, а підхід, архітектурний стиль проектування API.

В даній роботі мною був архітектурний стиль REST.

2.2 Характеристика архітектурного стилю REST

REST являє собою архітектурний стиль, який можна використовувати для створення програмних засобів, в яких клієнти (агенти користувачів) можуть відправляти запити службам (кінцевим точкам). REST є одним із способів реалізації архітектурного стилю «клієнт-сервер» - по суті, REST явно спирається на архітектурний стиль «клієнт-сервер».

Людина на ім'я Рой Томас Філдінг (Roy Thomas Fielding) першим ввів термін REST як концепції в своїй дисертації на степ доктора філософії «Архітектурні стилі і проектування архітектур програмних систем, що

підтримують роботу в мережі»). Він був одним з тих, хто працював над специфікацією, яка визначає сьогодні функціонування Інтернету: протоколом HTTP (Hypertext Transfer Protocol, протокол передачі гіпертексту). Зазвичай підготовка людей в області опису архітектурного стилю не має значення під час обговорення стилю, але в даному випадку, вважаю, це важливо, оскільки я вважаю, що для кращого розуміння основних принципів REST необхідно зрозуміти, що таке Інтернет і як він працює.

Можливо, Інтернет можна розглядати як найбільше, саме масштабування найпопулярніше додаток всіх часів. Обмеження стилю REST засновані на тих же базових принципах, які керують Інтернетом. Це такі принципи.

Агенти користувачів взаємодіють з ресурсами, якими може бути все, що можна перелічити і уявити. До кожного ресурсу можна звернутися за допомогою унікального ідентифікатора URI (Uniform Resource Identifier - універсальний код ресурсу).

Взаємодія з ресурсами (які виявляються за допомогою їх унікальних кодів URI) здійснюється за допомогою єдиного інтерфейсу стандартних команд HTTP (GET, POST, PUT і DELETE).

REST є основою сучасного Веб і являє собою формалізацію і узагальнення наявного стану речей, коли для передачі даних між клієнтом і сервером використовується протокол HTTPS, для ідентифікації ресурсів використовується механізм URI, а дані подаються в стандартизованих форматах. Архітектура REST визначається як безліч обмежень, що накладаються на деяку довільну многозвенную програмну архітектуру, при тому що деталі реалізації залишаються за рамками REST. Всі ці обмеження визначені.

REST є стиль створення Web API і сервісів, який повністю ґрунтується на HTTP. Ресурси, їх адреси URI, уявлення (XML, JSON і т.д.), Http Verbs, Headers, Reponse codes - все це складові частини REST, і дотримання стандартів і здоровому глузду, в поєднанні з цими віщим, допомагає нам створювати одноманітний інтерфейс (Uniform interface) для багатьох REST сервісів. REST має своє застосування і свою нішу. І так є ще такі речі як HATEOAS (Hyperme-

dia as the engine of application state), більш просунуте управління кешуванням, техніки авторизації, або такі які допомагають жити без сесії або навіть такі які допомагають робити транзакції, RSS / Atom, semantic web і ще багато іншого. Але все це лише продовження теми.

REST:

- Технологія створення веб-сервісів і web API
- Цільова аудиторія: публічний доступ, на противагу SOAP / WS - корпоративний сектор
- Statelessness, відсутність зберігання стану і сесій
- Багаторівнева архітектура з кешування з коробки (GET запити кешуються)
- Взаємодія з клієнтом через запит-відповідь, коди відповіді можуть сказати як пройшов запит
- Безпека через HTTPS і OAuth / OpenID
- Широкий спектр клієнтів на різних платформах і технологіях. За рахунок HTTP відмінно використовується з мобільних пристроїв. На рис. 2.1 зображена архітектура REST.

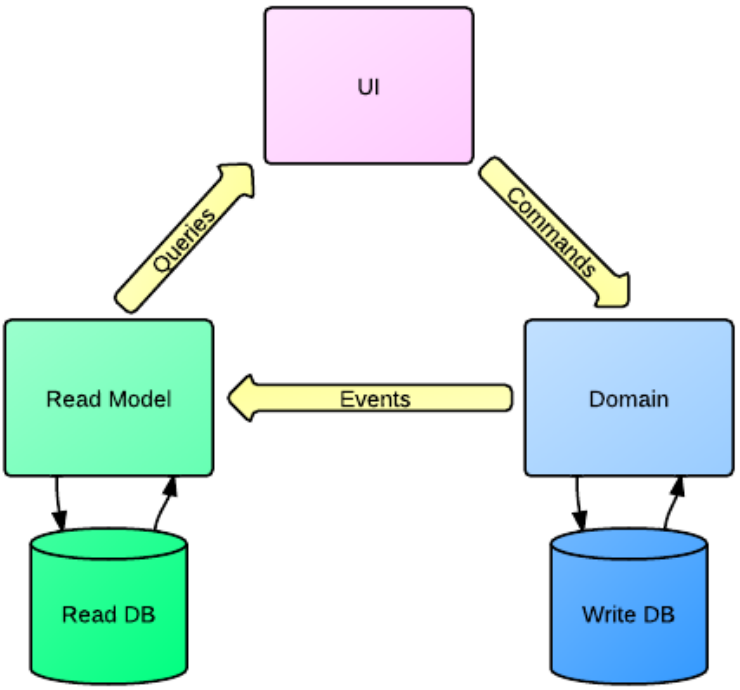


Рисунок 2.1 Відображення архітектури REST

2.3 Вибір метода для реалізації архітектури REST

Для реалізації REST було використано Spring Framework (або Spring, Весна) - це універсальна платформа з відкритим кодом для платформи Java. Також є роздріб для .NET-бази, що називається Spring.

Першу версію написав Род Джонсон, який вперше опублікував її виданням своєї книги «Експерт Один на один Java EE Design and Development» [5] (Vox Press, жовтень 2002).

Рамка була вперше випущена в червні 2003 року за ліцензією Apache 2.0. Перша стабільна версія 1.0 була випущена в березні 2004 року. Весна 2.0 вийшла у жовтні 2006 року, весна 2.5 у листопаді 2007 року, весна 3.0 у грудні 2009 року та весна 3.1 у грудні 2011 року. Поточна версія 5.2.4.

Хоча Біхар не надав жодної конкретної моделі програмування, вона поширилася в спільноті Java насамперед як альтернатива корпоративній моделі Java Benz. Весна забезпечує максимальну свободу розробникам Java для проектування. Крім того, він надає документацію та прості у використанні інструменти для вирішення проблем, що виникають при створенні корпоративних програм.

Тим часом функції Spring kernel застосовуються до будь-якої програми Java, і для розробки веб-додатків на корпоративній платформі Java існує безліч розширень та вдосконалень. З цих причин Біхар здобув величезну популярність і розробники визнані стратегічно важливими рамками.

Spring пропонує рішення багатьох проблем, які стоять перед розробниками Java та організаціями, які прагнуть будувати інформаційні системи на базі платформи Java. Через його широку функціональність важко визначити найважливіший структурний елемент, який він містить. Незважаючи на масштабну інтеграцію, Bihar не повністю інтегрована з платформою Java Enterprise, що є головною причиною її популярності.

Весна відома як потенційне джерело розширень (функцій), які можуть бути використані для ефективного розвитку складних бізнес-додатків далеко від важких моделей програмного забезпечення, що історично домінували в галузі. Домінували. Ще одна перевага полягає в тому, що вона впроваджує раніше не використовувані функціональні можливості в сьогоднішні домінуючі методи розвитку, навіть поза платформою Java

Цей фреймворк забезпечує постійну модель і стосується більшості типів додатків, вже побудованих на платформі Java. Вважається, що Біхар впроваджує модель розробки на основі кращих галузевих стандартів та робить її доступною у багатьох областях Java.

Spring має власну MVC-платформу веб-додатків, яка не була спочатку запланована. Розробники Spring вирішили написати її як реакцію на те, що вони сприйняли як невдалість конструкції (тоді) популярного Apache Struts, а також інших доступних веб-фреймворків. Зокрема, на їхню думку, не вистачило б поділ між шарами подання та обробки запитів, а також між шаром обробки запитів і моделлю. [6

Клас DispatcherServlet є основним контролером фреймворка і відповідає за делегування управління різних інтерфейсах, на всіх етапах виконання HTTP-запиту. Про ці інтерфейси слід сказати більш детально.

Як і Struts, Spring MVC є фреймворком, орієнтованим на запити. У ньому визначені стратегічні інтерфейси для всіх функцій сучасної запитання-орієнтованої системи. Мета кожного інтерфейсу - бути простим і зрозумілим, щоб користувачам було легко його заново імплементувати, якщо вони того побажають. MVC прокладає шлях до більш чистого front-end-коду. Всі інтерфейси тісно пов'язані з Servlet API. Цей зв'язок розглядається деякими як нездатність розробників Spring запропонувати для веб-додатків абстракцію більш високого рівня. Однак цей зв'язок залишає особливості Servlet API доступними для розробників, полегшуючи все ж роботу з ним. Найбільш важливі інтерфейси, певні Spring MVC, перераховані нижче:

HandlerMapping: вибір класу і його методу, які повинні обробити даний вхідний запит на основі будь-якого внутрішнього або зовнішнього для цього запиту атрибута або стану.

HandlerAdapter: виклик і виконання обраного методу обробки вхідного запиту.

Controller: включений між Моделлю (Model) і Виставою (View). Управляє процесом перетворення вхідних запитів в адекватні відповіді. Діє як ворота, направляючі всю інформацію, що надходить. Перемикає потік інформації з моделі в уявлення і назад.

View: відповідально за повернення відповіді клієнту у вигляді текстів і зображень. Деякі запити можуть йти прямо під View, не заходячи в Model; інші проходять через всі три шари.

ViewResolver: вибір, яке саме View має бути показано клієнту.

HandlerInterceptor: перехоплення вхідних запитів. Порівняємо, але не еквівалентний сервлет-фільтрам (використання не є обов'язковим і не контролюється DispatcherServlet-му).

LocaleResolver: отримання і, можливо, збереження локальних налаштувань (мова, країна, часовий пояс) користувача.

MultipartResolver: забезпечує Upload - завантаження на сервер локальних файлів клієнта. Spring MVC надає розробнику наступні можливості:

Ясна і прозоре поділ між шарами в MVC і запитах.

Стратегія інтерфейсів - кожен інтерфейс робить тільки свою частину роботи.

Інтерфейс завжди може бути замінений альтернативною реалізацією.

Інтерфейси тісно пов'язані з Servlet API.

Високий рівень абстракції для веб-додатків.

У веб-додатках можна використовувати різні частини Spring, а не тільки Spring MVC. На рис. 2.2 зображено ілюстрація модулів Spring.

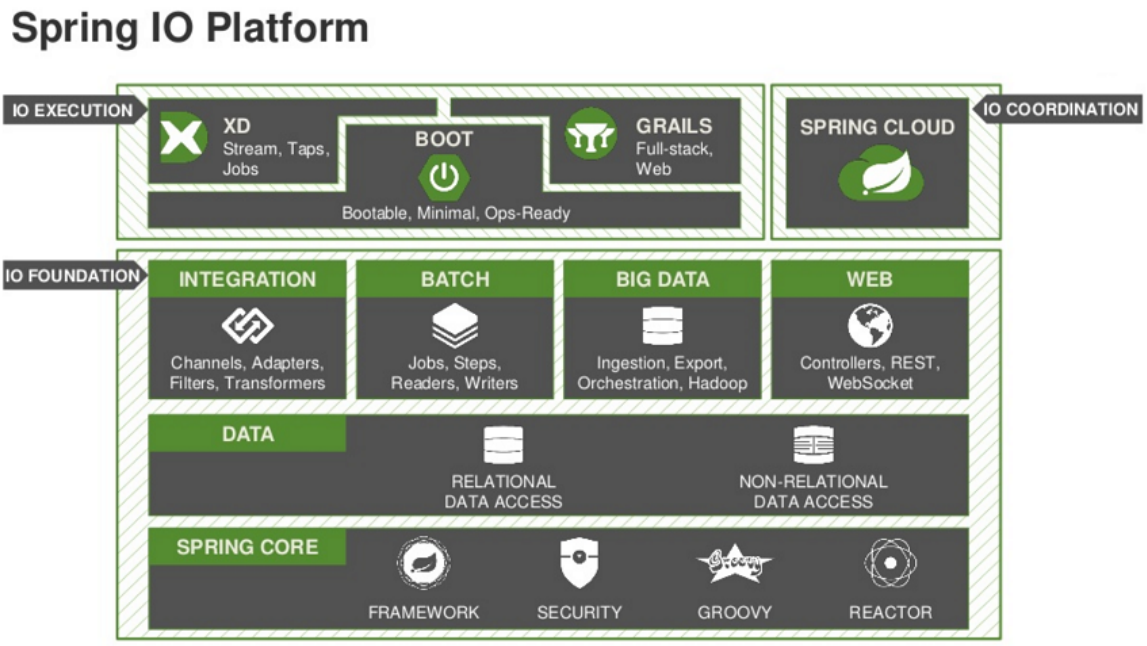


Рисунок 2.2 Ілюстрація модулів Spring

2.4 Архітектурні Переваги Spring framework

Spring може ефективно організувати об’єкти проміжного рівня, незалежно від того, чи вирішите ви використовувати EJB чи ні. Spring дбає про “склепання”, яке б залишилось на вашу долю, коли б ви використовували Struts чи інші фреймворки, залежних від певних J2EE APIs. А так як це, мабуть, найцінніше в проміжному рівні, Spring-ові сервіси управління конфігурацією можуть бути використані в будь-якому архітектурному рівні, в будь-якому runtime середовищі.

Spring може виключити неконтрольоване розмноження синглтонів, які спостерігаються в багатьох проектах. Насправді, це велика проблема, яка погано впливає на тестованість і об’єктну орієнтацію.

Spring може виключити необхідність в використанні різних форматів файлів для користувацьких властивостей, підтримуючи конфігурацію в несуперечливому стані для всіх застосувань і проектів. Вам траплялось цікавитись, які магічні ключі властивостей чи системні властивості необхідні конкретному класу, й лізти за цими відомостями в Javadoc чи навіть вихідний код. В Spring до-

сить подивитись у властивості JavaBean цього класу. Досягти такого спрощення допомагає використання Inversion of Control (інверсія керування, див. нижче).

Spring сприяє хорошому програмуванню, зменшуючи вартість програмування до інтерфейсів, замість класів, практично до нуля.

Spring спроектований так, щоб застосування, створені з його допомогою, залежали від найменшого можливого числа його API. Більшість бізнес-об'єктів в Spring-застосуваннях ніяк не залежать від Spring.

Для застосувань, створених за допомогою Spring, дуже просто проводити юніт-тестування.

Spring може зробити вибір EJB питанням реалізації, а не вирішуючим фактором архітектури застосування. Ви можете вирішити реалізовувати бізнес-інтерфейси як POJO (Plain Old Java Object, тобто прості об'єкти, а не Java Beans і т.п.) чи локальні EJB, ніяк не впливаючи цим на викликаний код.

Spring допомагає вирішити багато проблем, не втягуючи в це EJB. Spring може надати альтернативу EJB, застосовну для багатьох Web-застосувань. Наприклад, Spring може використовувати АОП для декларативного управління транзакціями без використання EJB-контейнера, і навіть без реалізації JTA, якщо обмежитись використанням одної БД.

Spring надає середовище доступу до даних з використанням JDBC чи таких ORM-продуктів, як Hibernate.

Spring надає стійку, просту модель програмування, перетворюючи її в ідеальний архітектурний "клей". Це можна простежити в підході Spring до JDBC, JMS, JavaMail, JNDI та багатьох інших важливих API.

2.5 Spring Security

Spring Security це Java / JavaEE framework, що надає механізми побудови систем аутентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework. Проект був розпочатий Беном Алексом (Ben Alex) в кінці 2003 року під

ім'ям «Acegi Security», перший реліз вийшов в 2004 році. Згодом проект був поглинений Spring'ом і став його офіційним дочірнім проектом. Вперше публічно представлений під новим ім'ям Spring Security 2.0.0 в квітні 2008 року.

Ключові об’єкти контексту

SecurityContextHolder, в ньому міститься інформація про поточний контексті безпеки програми, який включає в себе детальну інформацію про користувача (Principal) працює в даний час з додатком. За замовчуванням SecurityContextHolder використовує ThreadLocal для зберігання такої інформації, що означає, що контекст безпеки завжди доступний для методів виконуються в тому ж самому потоці. Для того що б змінити стратегію зберігання цієї інформації можна скористатися статичним методом класу SecurityContextHolder.setStrategyName (String strategy). Більш докладно SecurityContextHolder

SecurityContext, містить об'єкт Authentication і в разі необхідності інформацію системи безпеки, пов'язану із запитом від користувача.

Authentication представляє користувача (Principal) з точки зору Spring Security.

GrantedAuthority відображає дозволу видані користувачу в масштабі всього програми, такі дозволи (як правило називаються «ролі»), наприклад ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN.

UserDetails надає необхідну інформацію для побудови об'єкта Authentication з DAO об'єктів додатки або інших джерел даних системи безпеки. Об'єкт UserDetailscодержіт ім'я користувача, пароль, прапори: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled і Collection - прав (ролей) користувача.

UserDetailsService, використовується щоб створити UserDetails об'єкт шляхом реалізації єдиного методу цього інтерфейсу

UserDetails loadUserByUsername (String username) throws UsernameNotFoundException;

Дозволяє отримати з джерела даних об'єкт користувача та сформувати з нього об'єкт UserDetails який буде використовуватися контекстом Spring Security. На рис. 2.3 ілюстровано модуля Spring Security.

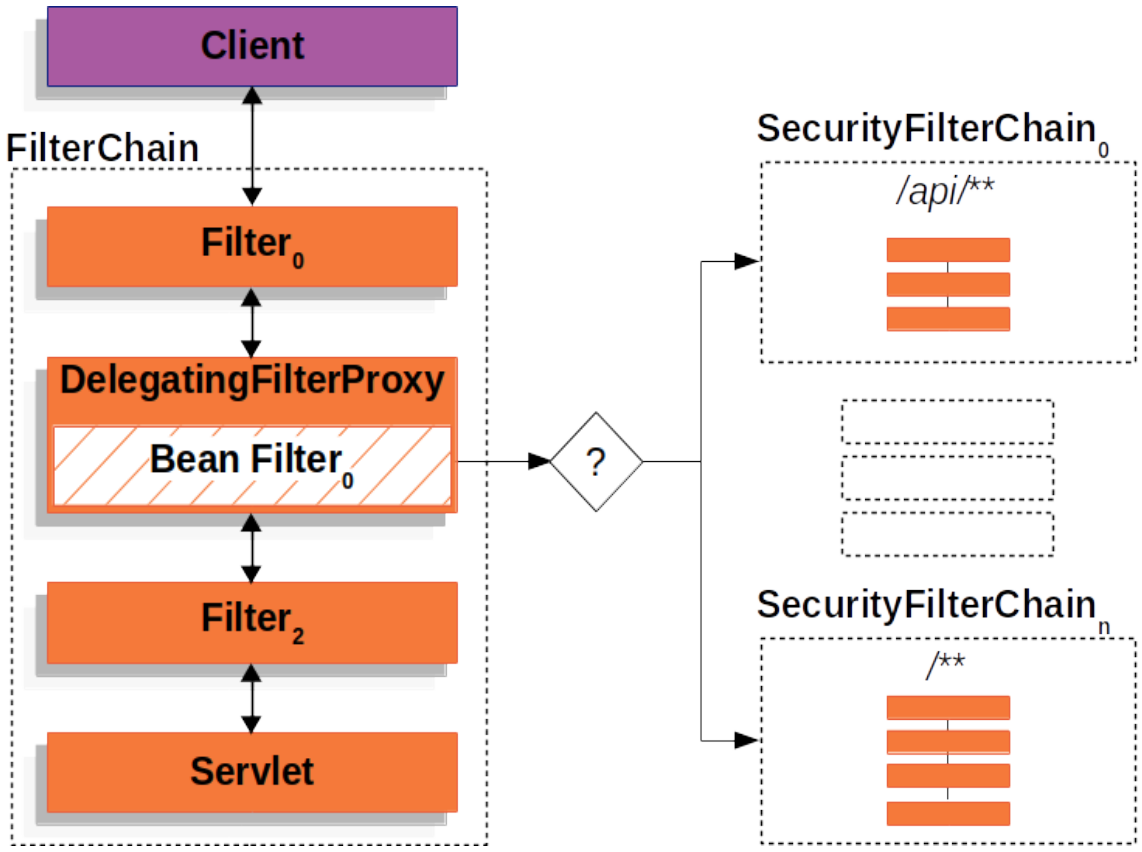


Рисунок 2.3 Ілюстрація модуля Spring Security

2.5.1 Аутентифікація

Користувачеві буде запропоновано увійти в систему надавши ім'я (логін або email) і пароль. Ім'я користувача і пароль об'єднуються в екземпляр класу UsernamePasswordAuthenticationToken (екземпляр інтерфейсу Authentication) після чого він передається примірнику AuthenticationManager для перевірки.

У випадку якщо пароль не збігається з назвою користувача буде викинуто виключення BadCredentialsException з повідомленням "Bad Credentials".

Якщо аутентифікація пройшла успішно повертає повністю заповнений примірник Authentication.

Для користувача встановлюється контекст безпеки шляхом виклику методу SecurityContextHolder.getContext (). SetAuthentication (...), куди передається об'єкт який повернув AuthenticationManager. На рис. 2.4 ілюстровано аутентифікації за допомогою Spring Security.

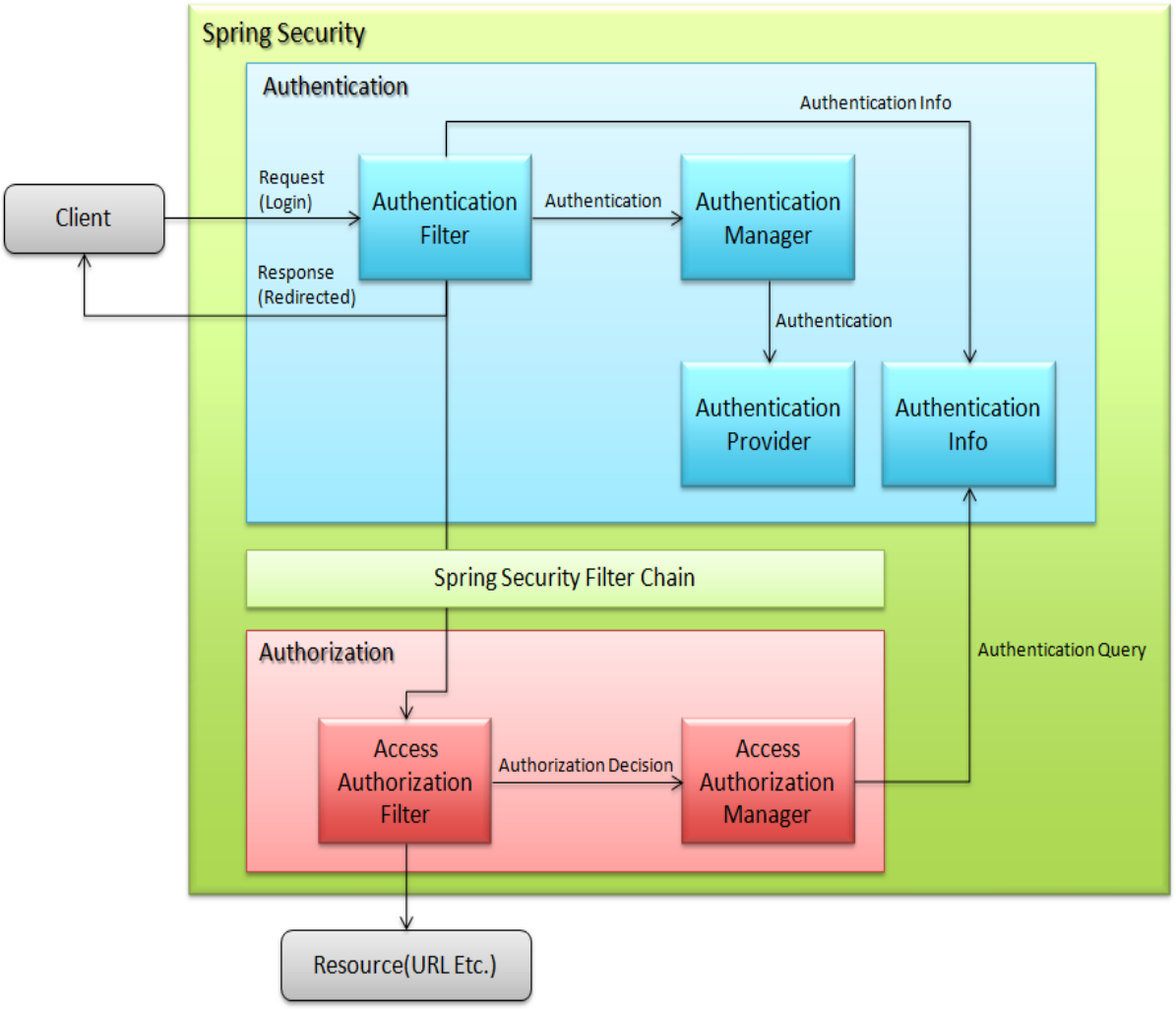


Рисунок 2.4 Ілюстрація аутентифікації за допомогою Spring Security

2.6 Вибір бази даних

Для вирішення проблематики, було вирішено використовувати Postgre-SQL.

Модель даних

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird.

Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить Постгрес неймовірно гнучким і надійним. Серед іншого, він вміє створювати, зберігати та видавати складні структури даних. У деяких прикладах нижче ви побачите вкладені і складові конструкції, які не підтримуються стандартними РСУБД.

Структури і типи даних

Існує великий список типів даних, які підтримує Постгрес. Крім числових, з плаваючою точкою, текстових, булевих і інших очікуваних типів даних (а також безлічі їх варіацій), PostgreSQL може похвалитися підтримкою uuid, грошового, що перераховується, геометричного, бінарного типів, мережевих адрес, бітових рядків, текстового пошуку, xml, json , масивів, композитних типів і діапазонів, а також деяких внутрішніх типів для ідентифікації об'єктів і розташування балок. Справедливості заради варто сказати, що MySQL, MariaDB і Firebird теж мають деякі з цих типів даних, але тільки Постгрес підтримує їх все.

Підтримка JSON

Підтримка JSON в PostgreSQL дозволяє вам перейти до зберігання schema-less даних в SQL базі даних. Це може бути корисно, коли структура даних вимагає певної гнучкості: наприклад, якщо в процесі розробки структура все ще змінюється або невідомо, які поля буде містити об'єкт даних.

Тип даних JSON забезпечує перевірку коректності JSON, який дозволяє використовувати спеціалізовані JSON оператори і функції, вбудовані в Postgre-SQL для виконання запитів і маніпулювання даними. Також доступний тип JSONB - двійкова різновид формату JSON, у якій прогалини видаляються, сортування об'єктів не зберігається, натомість вони зберігаються найбільш оптимальним чином, і зберігається тільки останнє значення для ключів-

дублікатів. JSONB зазвичай є кращим форматом, оскільки вимагає менше місця для об'єктів, може бути проіндексовані і обробляється швидше, так як не вимагає повторного синтаксичного аналізу.

В MySQL 5.7.8 і MariaDB 10.0.1 була додана підтримка вбудованих об'єктів JSON. Але, хоча існує безліч функцій і операторів для JSON, які тепер доступні в цих базах даних, вони не індексуються так, як JSONB в PostgreSQL. Firebird поки що не долучився до тренду і підтримує об'єкти JSON тільки у вигляді тексту.

Цілісність даних

PostgreSQL прагне відповідати стандарту ANSI-SQL: 2008, відповідає вимогам ACID (атомарность, узгодженість, ізольованість і надійність) і відомий своєю посилальної і транзакционной цілісністю. Первинні ключі, що обмежують і каскадні зовнішні ключі, унікальні обмеження, обмеження NOT NULL, перевірочні обмеження та інші функції забезпечення цілісності даних дають впевненість, що тільки коректні дані будуть збережені.

MySQL і MariaDB більше працюють на те, щоб відповідати стандарту SQL з двигунами таблиць InnoDB / XtraDB. Тепер вони пропонують опцію STRICT з використанням режимів SQL, яка встановлює перевірки коректності використовуваних даних. Незважаючи на це, в залежності від того, який режим ви використовуєте, недостовірні і навіть урізані без вашого відома дані можуть бути вставлені або створені при оновленні. Жодна з цих баз даних зараз не підтримує CHECK обмеження. Крім того, у них існує безліч особливостей щодо обмежень посилальної цілісності по зовнішнім ключам. На додаток до вищесказаного, цілісність даних може істотно постраждати в залежності від обраного движка зберігання. MySQL (і fork MariaDB) не роблять секрету з того, що проміняли цілісність і відповідність стандартам на швидкість і ефективність.

Вердикт

У PostgreSQL безліч можливостей. Створений з використанням об'єктно-реляційної моделі, він підтримує складні структури і широкий спектр вбудованих і обумовлених користувачем типів даних. Він забезпечує розширену ємність даних і заслужив довіру дбайливим ставленням до цілісності даних. Він має новітні функції зберігання даних.

2.7 Вибір технологій для frontend

Для написання frontend часини було використано бібліотеку vue.js через ряд своїх переваг.

Vue.js - це JavaScript бібліотека для створення веб-інтерфейсів з використанням шаблону архітектури MVVM (Model-View-ViewModel).
Оскільки Vue працює тільки на «рівні уявлення» і не використовується для проміжного програмного забезпечення і бекенда, він може легко інтегруватися з іншими проектами і бібліотеками. Vue.js містить широку функціональність для рівня уявлень і може використовуватися для створення потужних односторінкових веб-додатків.

Функції Vue.js:

- Реактивні інтерфейси;
- Декларативний рендеринг;
- Зв’язування даних;
- Директиви (всі директиви мають префікс «V-». В директиву передається значення стану, а в якості аргументів використовуються html атрибути або Vue JS події);
- Логіка шаблонів;
- Компоненти;
- Обробка подій;
- Переходи і анімація CSS;
- Фільтри.

Основна бібліотека Vue.js 2 дуже маленька (всього 17 кБ). Це гарантує, що навантаження на ваш проект, реалізований за допомогою Vue.js, мінімальна, а ваш сайт буде швидко завантажуватися. Завантажити відповідний .js файл можна за посиланням.

РОЗДІЛ 3 ОПИС РОЗРОБЛЮВАЛЬНОЇ ПРОГРАМИ

3.1 Опис ролей в системі та їх функцій

В системі будуть присутні такі ролі

- Роль користувача (студента)
- Роль тренера
- Роль адміністратора

Для різних ролей буде доступний різний функціонал, зокрема звичайний користувач зможе мати можливість користватися функціями яки йому призначені. Тренер буде мати доступ до функціоналу який доступний звичайному користувачу, також він буде отримувати доступ до функціоналу призначеного для тернера.

Адміністратор має повні права змінюваті бкдь-які дані, і доступ до використання функціналу тренера, також до функціоналу призначеного для адміністрування.

Функціонал для звичайного користувача:

- Можливість авторизуватися, зареєструватися.
- Переглядати інформацію щодо секцій, філіалів
    - Розклад занять
    - Інформація про локацію проведення заняття
    - Інформація про тренерів
    - Перегляд фото та відео
- Перегляд новин
- Перегляд публічних сторінок тренерів та студентів
- Загальний розклад тренувань
- Інформація щодо турнів та змагань
- Інформація про доступні абонементи
- Можливість купівлі абонемента
- Можливість створення власної публічної сторінки
- Можливість запису на груупові та індивідуальні заняття

- Можливість перегляду свого влсного розкладу
- Можливість перенесення заняття
- Можливість завантаження відео та фото
- Можливість залишити заяви
- Можливість залишати оцінки та коментарії щодо тренування
- Отримання сповіщення

Функціонал для тернер:

- Фукціонал, яки доступний для звичайного користувача
- Перегляд свого графіку роботу
- Перегляд статистики роботи
    - Кількість годин
    - Оцінки, рейтинг
    - Кількість студентів
- Можливість зміни графіку роботи
- Звязок з адміністратором
- Можливість введення блогу
- Введеня звітності
    - Можливість відмічати кількість присутніх студентів
    - Можливість відмічати оплату студента
    - Можливість залишати коментарії щодо спортивного інвертарю
    - Можливість відмічати витрати
- Надавання студентам привілегії, зміння рангової системи
- Можливість отримати сповіщення з інформацією про:
    - Заяви
    - Найближчі тренування
    - Коментарії
    - Зміни в розкладі

Функціонал для адміністратора:

- Функціонал, який доступний для тренера

- Робота та внесенні змін до різних модулів
    - Новини
    - Абонементи
    - Групові та індивідуальні заняття
    - Інформація щодо внутрішніх проектів
    - Інформація про турніри
    - Філіали та секції
    - Розклад
    - Фото та відео
    - Контактної інформації
    - Нагороди для користувачів
- Можливість зміни данних користувача, тренера
- Прийняття та обровка заяв
- Статистика щодо роботи:
    - Філіалів
    - Секцій
    - Тренерів
    - Спортсменів
- Контроль контенту розміщеного користувачем, тренером
- Можливість запису користувача на заняття
- Фінансова статистика та аналітика

3.2 Реалізація доступу до функціоналу різних ролей

Різні ролі було добавлено за допомогою модуля Spring Security. Зокрема було використано стандарт JSON Web Token та за допомгою філтрів з рошрирення стандартого класу Spring Security WebSecurityConfigurerAdapter.

Приклад реалізацію конфігураційного класу для налаштування фільтрів запитів

```
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final JwtTokenProvider jwtTokenProvider;
    private static final String ADMIN_ENDPOINT = "/api/v1/admin/**";
    private static final String LOGIN_ENDPOINT = “/api/v1/auth/login";

    @Autowired
    public SecurityConfig(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }
    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .httpBasic().disable()
                .csrf().disable()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.S-
TATELESS)
                .and()
                .authorizeRequests()
                .antMatchers(LOGIN_ENDPOINT).permitAll()
                .antMatchers(ADMIN_ENDPOINT).hasRole("ADMIN")
                .anyRequest().authenticated()
                .and()
                .apply(new JwtConfigurer(jwtTokenProvider));
```

```
    }
}
```

Також було реалізовано основні класи для формування та загалом циклу життя токену

JWTConfigurer

```
public class JwtConfigurer extends SecurityConfigurerAdapter<DefaultSecurityFilterChain, HttpSecurity> {
    private JwtTokenProvider jwtTokenProvider;

    public JwtConfigurer(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @Override
    public void configure(HttpSecurity httpSecurity) throws Exception {
        JwtTokenFilter jwtTokenFilter = new JwtTokenFilter(jwtTokenProvider);
        httpSecurity.addFilterBefore(jwtTokenFilter, UsernamePasswordAuthenticationFilter.class);
    }
}
```

JWTTokenFilter

```
public class JwtTokenFilter extends GenericFilterBean {

    private JwtTokenProvider jwtTokenProvider;

    public JwtTokenFilter(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain filterChain)
            throws IOException, ServletException {

        String token = jwtTokenProvider.resolveToken((HttpServletRequest) req);
        if (token != null && jwtTokenProvider.validateToken(token)) {
            Authentication auth = jwtTokenProvider.getAuthentication(token);
```

```
            if (auth != null) {
                SecurityContextHolder.getContext().setAuthentication(auth);
            }
        }
        filterChain.doFilter(req, res);
    }

}
```

JWTTokenProvider

```
@Component
public class JwtTokenProvider {

    @Value("${jwt.token.secret}")
    private String secret;

    @Value("${jwt.token.expired}")
    private long validityInMilliseconds;


    @Autowired
    private UserDetailsService userDetailsService;

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
        return bCryptPasswordEncoder;
    }

    @PostConstruct
    protected void init() {
        secret = Base64.getEncoder().encodeToString(secret.getBytes());
    }

    public String createToken(String username, List<Role> roles) {

        Claims claims = Jwts.claims().setSubject(username);
        claims.put("roles", getRoleNames(roles));
```

```
        Date now = new Date();
        Date validity = new Date(now.getTime() + validityInMilliseconds);

        return Jwts.builder()//
                .setClaims(claims)//
                .setIssuedAt(now)//
                .setExpiration(validity)//
                .signWith(SignatureAlgorithm.HS256, secret)//
                .compact();
    }

    public Authentication getAuthentication(String token) {
        UserDetails userDetails = this.userDetailsService.loadUserByUsername(get-
Username(token));
        return new UsernamePasswordAuthenticationToken(userDetails, "", userDetail-
s.getAuthorities());
    }…
```

- **JWTUser** (опис таблиці бази даних у вигляді класу)
- **JWTUserDetaileService**
- **JWTUserFactory**

На рис. 3.1 зображено діаграму зв’язків таблиць.

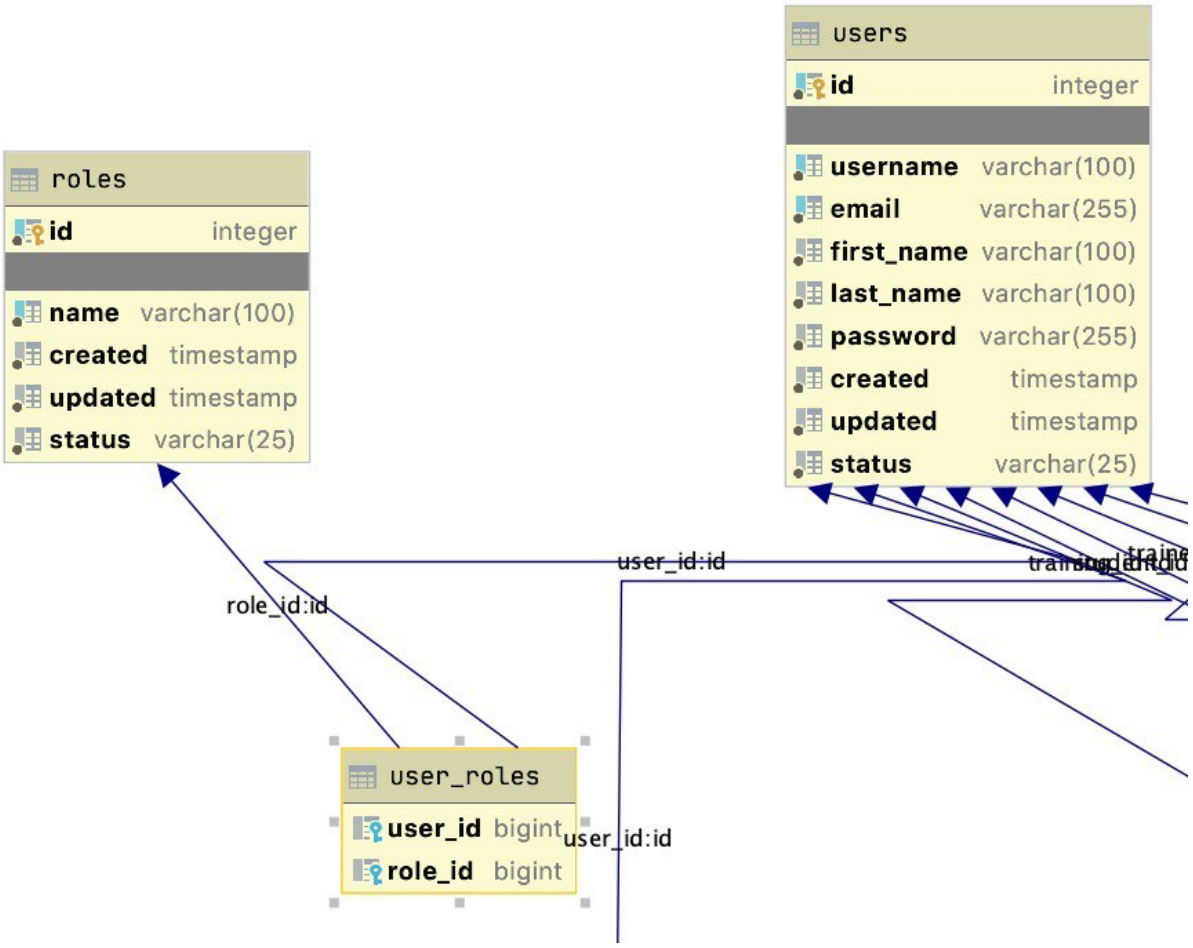


Рисунок 3.1 – Діаграма зв'язків таблиці

3.3 Реалізація шару роботи з базою данних

Механізм управління даними реалзіований за допомогою DAO та **Crud-Repository**(інтерфейс модуля Spring Data).

DAO (data access object) – інтерфейс до механізму зберігання даних в базі даних.

Задля зручності управління даними були спроектовані класи такого ж типу як і таблиці.

Приклад моделі таблиці User

```
@Entity
@Table(name = "users")
@Data
```

```
public class User extends BaseEntity {

    @Column(name = "username")
    private String username;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "email")
    private String email;

    @Column(name = "password")
    private String password;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "user_roles",
            joinColumns = {@JoinColumn(name = "user_id", referencedColumnName =
"id")},
            inverseJoinColumns = {@JoinColumn(name = "role_id", referencedColumn-
Name = "id")})
    private List<Role> roles;
}
```

Задля зручності обробки запиту було використано **DTO** (Data Transfer Object) – шаблон для передачі даних між підсистемами.

```
public static TrainingDto fromTrainingAdmin(Training training){
        TrainingDto trainingDto = new TrainingDto();
        trainingDto.setId(training.getId());
        trainingDto.setName(training.getName());
        trainingDto.setBranchId(training.getBranch().getId());

        trainingDto.setHallId(training.getHall().getId());

        trainingDto.setCapacity(training.getCapacity());
        trainingDto.setTime(training.getTime());

        trainingDto.setCreated(training.getCreated());
        trainingDto.setUpdated(training.getUpdated());
        trainingDto.setStatus(training.getStatus());

        trainingDto.setTrainingType(training.getTrainingType());
        return trainingDto;
    }

    public static TrainingDto fromTrainingAdminWithTrainer(Training training){
        TrainingDto trainingDto = fromTrainingAdmin(training);
        List<User> list =  training.getTrainer();
        List<Long> ids = new ArrayList<>();
        for(User user:list){
            ids.add(user.getId());
        }
        trainingDto.setTrainerIds(ids);
        return trainingDto;
    }
```

```
    public static TrainingDto fromTrainingWithTrainer(Training training){
        TrainingDto trainingDto = fromTraining(training);
        List<User> list =  training.getTrainer();
        List<Long> ids = new ArrayList<>();
        for(User user:list){
            ids.add(user.getId());
        }
        trainingDto.setTrainerIds(ids);
        return trainingDto;
    }

    public static TrainingDto fromTraining(Training training){
        TrainingDto trainingDto = new TrainingDto();
        trainingDto.setId(training.getId());
        trainingDto.setName(training.getName());
        trainingDto.setBranchId(training.getBranch().getId());
        trainingDto.setHallId(training.getHall().getId());
        trainingDto.setCapacity(training.getCapacity());
        trainingDto.setTime(training.getTime());
        trainingDto.setTrainingType(training.getTrainingType());
        return trainingDto;
    }

    public static ArrayList<TrainingDto> fromArrayTraining(List<Training> train-
ingsList){
        ArrayList<TrainingDto>  trainingDtoArrayList = new ArrayList<>();
        for (Training training:trainingsList){
            trainingDtoArrayList.add(fromTrainingWithTrainer(training));
        }
        return trainingDtoArrayList;
```

```
    }

    public static ArrayList<TrainingDto> fromArrayTrainingAdmin(List<Training>
trainingsList){
        ArrayList<TrainingDto>  trainingDtoArrayList = new ArrayList<>();
        for (Training training:trainingsList){
            trainingDtoArrayList.add(fromTrainingAdminWithTrainer(training));
        }
        return trainingDtoArrayList;
    }

    public Training toTrainingAdmin(){
        Training training = new Training();
        //training.setId(id);
        training.setName(name);

        Hall hall = new Hall();hall.setId(hallId);
        training.setHall(hall);
        // training.setLocation(location);
        Branch branch = new Branch();branch.setId(branchId);
        training.setBranch(branch);
        training.setCapacity(capacity);
        training.setTime(time);

        ArrayList<User> arrayListTrainer = new ArrayList<>();
        for(Long id:trainerIds){
            User user = new User();
            user.setId(id);
            arrayListTrainer.add(user);
        }
```

```
        training.setTrainer(arrayListTrainer);



        training.setStatus(status);

        training.setTrainingType(trainingType);
        return training;
    }

    public Training toTrainingAdminWithId(){
        Training training = new Training();
        training.setId(id);
        training.setName(name);
        Hall hall = new Hall();hall.setId(hallId);
        training.setHall(hall);
        Branch branch = new Branch();branch.setId(branchId);
        training.setBranch(branch);
        training.setCapacity(capacity);
        training.setTime(time);
        ArrayList<User> trainerList = new ArrayList<>();
        for(Long id: trainerIds){
            User trainer = new User();
            trainer.setId(id);
            trainerList.add(trainer);
        }
        training.setTrainer(trainerList);
        training.setStatus(status);

        training.setTrainingType(trainingType);
        return training;
```

```
    }
}
```

3.4 База даних

В якості бази даних було використано PostgresSQL. Також було використано бібліотеку liquidbase задля простого перенесня структури БД на любий комп’ютер, сервер та для контролю логування змін внесених до БД. На рис. 3.2 зображено даіграму БД.

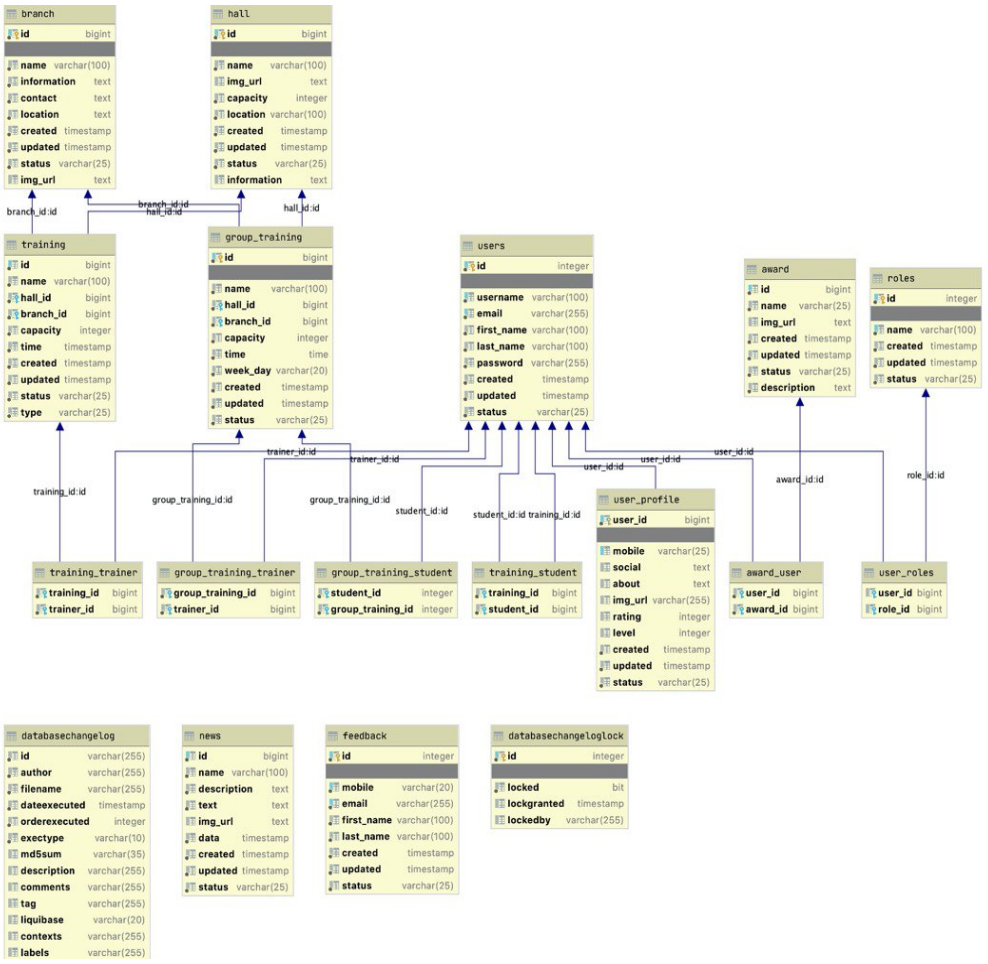


Рисунок 3.2 – Діаграма БД

3.5 Результати роботи API

На рис. 3.3 Зображено реалізацію веб-інтерфейсу.

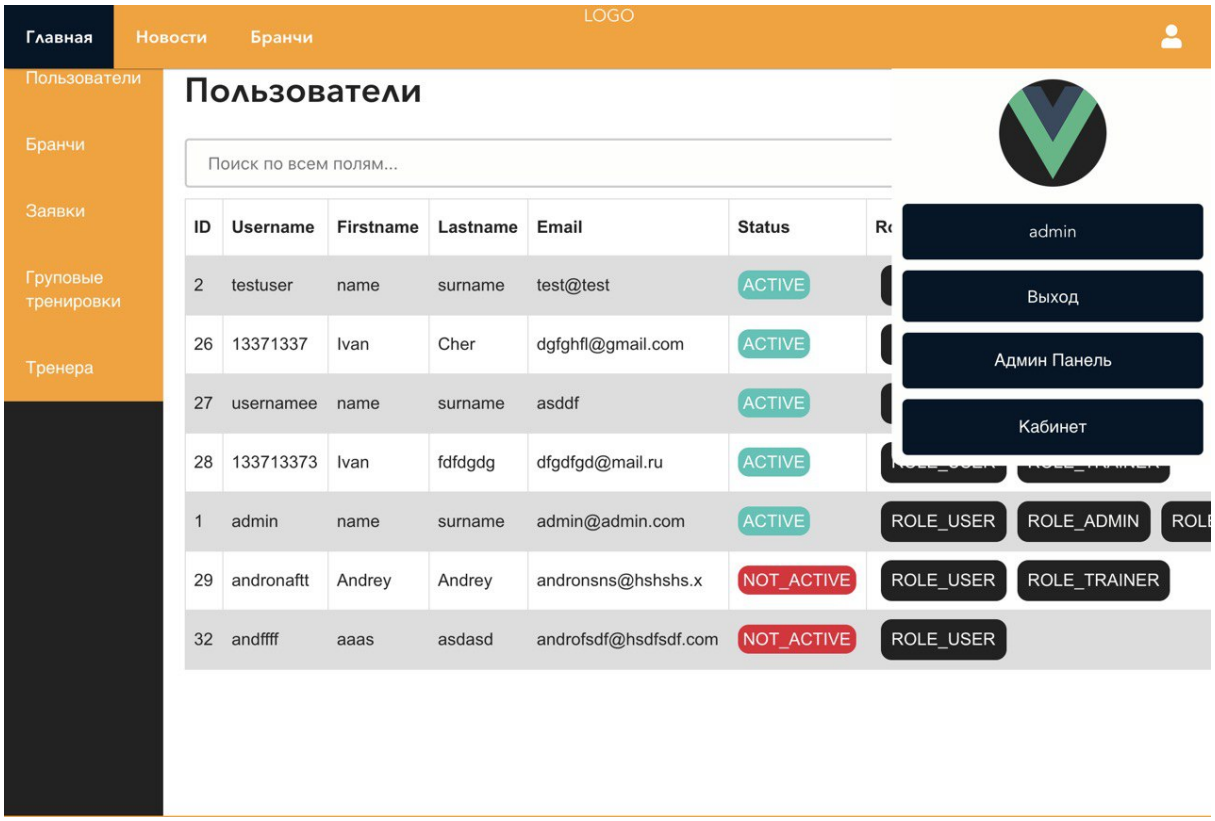


Рисунок 3.3 – Реалізація веб-інтерфейсу

Веб- інтерфейс доступний за посиланням -http://teamdancecommuni-ty.herokuapp.com

Посилання для доступу до API - https://team-dance-community.herokuapp.com

Посилання з описом API, доступне в програмі Katalon https://github.com/andronaft/Team_Dance_Community/tree/katalon

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі

Спроектувати та реалізувати програмний продукт для автоматизації адміністрування та спрощення адміністративних справ в сфері надання спортивних послуг. ПП може бути використаним на будь-якій платформі, зокрема, для початку буде реалізовано веб-версію, у подальшому можливий перехід на різні платформи.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, а саме системи автоматизації адміністрування у вигляду RESTсервісу. Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

F_1 – вибір мови програмування;

F_2 – вибір оптимальної СКБД;

F_3 – інтерфейс користувача.

Функції мають варіанти.

Функція F_1:

а) мова програмуванняC#;

б) мова програмування Java;

Функція F_2:

а) MySQL;

б)PostgreSL.

Функція F_3:

а) інтерфейс користувача, створений за технологією ASPNet;

б) інтерфейс користувача, створений за технологією Spring.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну

матрицю варіантів основних функцій (таблиця 4.1).

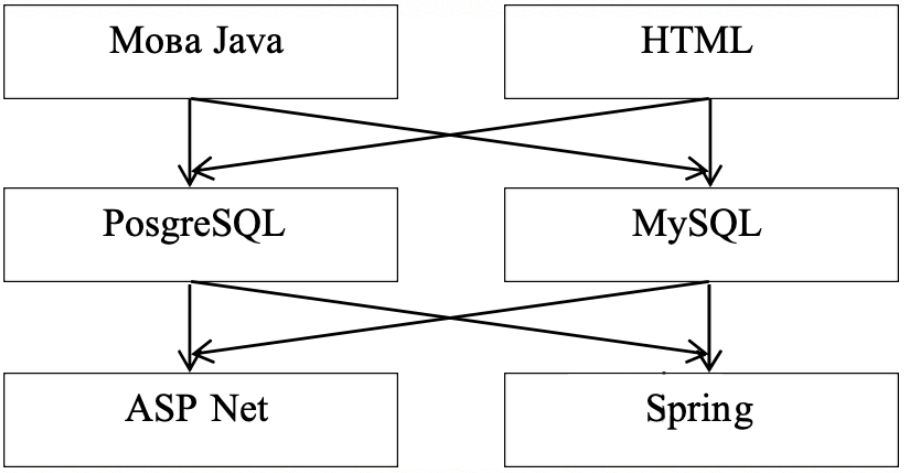


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Осноні функції	Варіани реалізації	Переваги	Недоліки
F1	А	Кросплатформений	Низька швидкодія
	Б	Займає менше часупри написанні коду	Більшийчаснавиконання операцій
F2	А	Безкоштовність	Відсутність вкладених запитів
	Б	Надійність, внесення змін безперезапуску, безкоштовність	Необхідність додаткової інсталяції, низький рівень користувацької підтримки
F3	А	Простота створення	Відсутність кросплатформеності
	Б	Простота створення,	Кросплатформеність

На основі аналізу позитивно-негативної матриці робимо висновок, при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічнійкарті.

Функція *F1:*

Оскільки розрахунки проводяться з великими об'ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант а) має бути відкинутий.

Функція *F2:*

Вибір СКБД відіграє велику роль у даному програмно мупродукті, тому вважаємо варіант а) оскільки, PostgresSQL краще підходить для вирішення данної проблематики.

Функція *F3:*

Вибір мови програмування не є настільки важливим. Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1б – F2а –F3а
2. F1б – F2а –F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Длятого, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

− *X1* – швидкодія мовипрограмування, параметр функції F1;

− *X2* – об’єм пам’яті для збереженняданих, параметр функції F2;

− *X3* – час обробкиданих, параметр функції F3;

− *X4* – потенційний об’єм програмного коду, параметр функції F3.

4.3 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл.4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	18000	10000	1000
Об’ємпам’яті для збереження даних	X2	Мб	28	14	7
Час обробки запитів користувача	X3	мс	1200	520	100
Потенційний об’єм програмного коду	X4	кількість строк коду	2200	1600	900

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

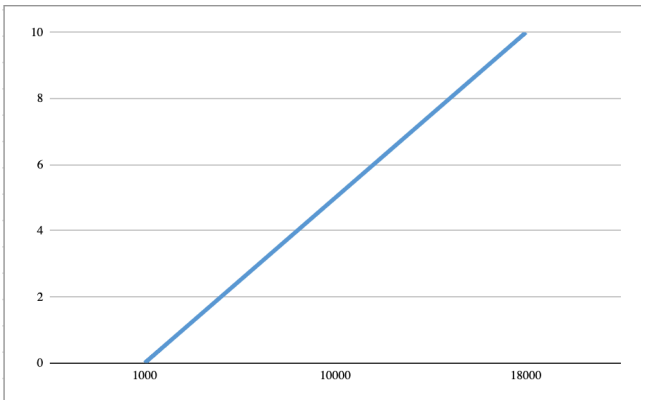


Рисунок 4.2 – X1, швидкодія мови програмування

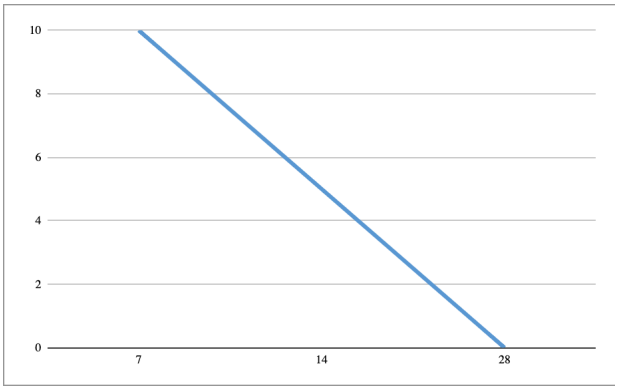


Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

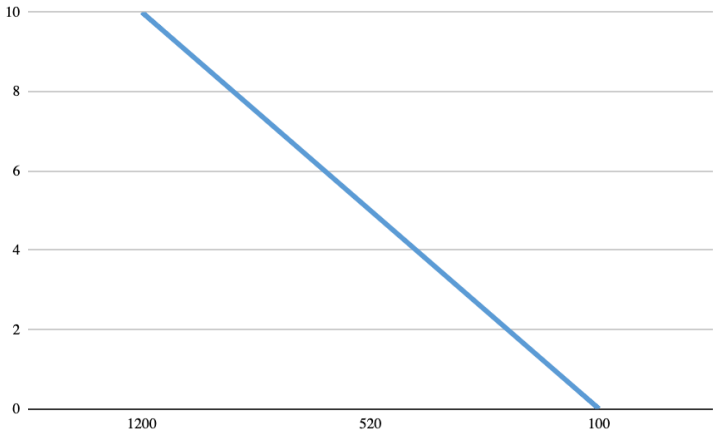


Рисунок 4.4 – X3, час виконання запитів користувача

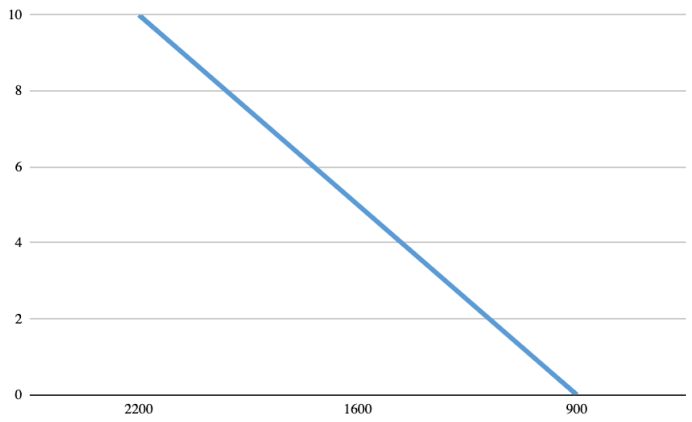


Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметру	Одиниці виміру	Ранг параметраза оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	1	1	1	1	1	1	7	-10,5	110,25
X2	Об’ємпам’яті для збереження даних	Мб	2	3	2	4	2	2	2	17	-0,5	0,25
X3	Часобробки запитів користувача	Мс	3	2	3	3	3	3	4	21	3,5	12,25

X4	Потенційний об'єм програмного коду	Кількість строк коду	4	4	4	2	4	4	3	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	179

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = 0.73 > W_k = 0{,}67.$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.
Скориставшись результатом ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів –

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 та X_2	>	>	>	>	>	>	>	>	1.5
X_1 та X_3	>	>	>	>	>	>	>	>	1.5
X_1 та X_4	>	>	>	>	>	>	>	>	1.5
X_2 та X_3	>	<	>	<	>	>	>	>	1.5
X_2 та X_4	>	>	>	<	>	>	>	>	1.5
X_3 та X_4	>	>	>	<	>	>	<	>	1.5

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри	Параметри				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X_1	1,00	1,50	1,50	1,50	5,50	0,32352	22,75	0,3345	90,625	0,3353
X_2	1,50	1,00	1,50	1,50	5,50	0,32352	22,75	0,3345	90,625	0,3353
X_3	0,50	0,50	1,00	1,50	3,50	0,20588	12,75	0,1875	50,125	0,1854
X_4	0,50	0,50	0,50	1,00	2,50	0,14705	9,75	0,1433	38,875	0,1438
Загалом:					17,00	1,00	68,00	1,00	270,25	1,00

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

F1б – F2а –F3а

F1б – F2а –F3б

Таблиця 4.6 – Розрахунок коефіцієнту якості рівняння

Основні функції-	Варіант реалізації функції	Абсолютне значення параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	Б	4200	2	0,3353
F2(X2)	А	22.4	2	0,3353
F3(X3,X4)	А - X3	352	3	0,1854
	А - X4	1180	2	0,1438
	Б - X3	268	2	0,1854
	Б - X4	1040	1	0,1438

За даними з таблиці 4.6 за формулою визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,6706 + 0,6706 + 0,5562 + 0,2876 = 2,185

K_{K2} = 0,6706 + 0,6706 + 0,3708 + 0,1438 = 1,8558

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;
3. Реалізація алгоритму, що виводить результат з графічною
4. Реалізація алгоритму, виводу тексту на екран програмного продукту

Варінат 1 містить ще одне завдання:

5. Реалізація фільтрів Https запиту.

Варіант 2 містить ще одне завдання:

6. Реалізація авторизації за допомогою Spring security.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б, завдання 3, 4 – до групи В. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3, завдання 3 – до групи 2, а завдання 4 - до групи 3.

Для першого завдання (степінь новизни А, група складності – 1), тобто T_P=90 людино-днів, $K_П$ = 1.7, $K_{СК}$ = 1; $K_{СТ}$ = 0.8.

$T_1 = 90 \cdot 0.8 \cdot 1.7 = 122.4$ людино-днів.

Для другого завдання (степінь новизни Б, група складності – 3), тобто $T_Р$=19 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

T_2 = 19 · 0.8 ·0.9 = 13.68 людино-днів.

Для третього завдання (степінь новизни В, група складності – 2), тобто $T_Р$=19 людино-днів, $К_П$ = 1.2; $К_{СК}$ = 1; $К_{СТ}$ = 0.6:

T_3 = 19·1.2 ·1· 0.6 = 13.68 людино-днів.

Для четвертого завдання (степінь новизни В, група складності – 3),тобто $T_Р$=12 людино-днів,$К_П$ = 0.8; $К_{СК}$ = 1; $К_{СТ}$ = 0.6:

T_4 = 12 · 0.8·1· 0.6 = 5.76 людино-днів.

Складаємо трудомісткість завдання для кожного з обраних варіантів реалізації програми:

T_I = (122.4 + 13.68 + 13.68) ·6 = 898.56 людино-годин;

T_{II} = (122.4 + 13.68 + 5.76) ·6 = 851.04 людино-годин;

Більш високу трудомісткість має варіант І.

В розробці бере участь один програміст з окладом 13000 грн. Визначимо зарплату за годину за формулою:

$$СЧ = \frac{13000}{1 \cdot 21 \cdot 8} = 77{,}38\ грн.$$

Зарплата розробників за варіантами становить:

I. $С_{ЗП1}$ = 77.38 · 898.56 · 1.2 = 83436,68грн.

II. $С_{ЗП2}$ = 77.38 · 851.04 · 1.2 = 79024.17 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД1}$ = $С_{ЗП}$ · 0.22= 83436,68 · 0.22 = 18356.06 грн.

II. $С_{ВІД2}$ = $С_{ЗП}$ · 0.22= 79024.17 · 0.22 = 17385.31 грн.

Тепер визначимо витрати на оплату однієї машино-години ($С_М$). Так як одна ЕОМ обслуговує одного програміста з окладом 13000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 13000 \cdot 0.2 = 31200$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 31200 \cdot (1 + 0.35) = 42120$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 42120 \cdot 0.22 = 9266.4$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 12000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 12000 = 3450$ грн.,

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 12000 \cdot 0.05 = 690$ грн.,

Ефективний годинний фонд часу персонального комп'ютеру за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (366 – 118 – 16) \cdot 8 \cdot 0.9 = 1670.4$ годин,

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1670.4 \cdot 0.156 \cdot 0.9733 \cdot 1.75 = 443.73$ грн.,

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 12000 \cdot 0.67 = 8040$ грн.

Тоді, річні експлуатаційні витрати будуть:

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$

$С_{ЕКС} = 42120 + 9266.4 + 3450 + 690 + 443.73 + 8040 = 64010.13$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 64010.13/1670.4 = 38.32$ грн/час.

Витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

I. $С_{М1} = С_{М\text{-}Г} \cdot Т_1 = 38.32 \cdot 898.56 = 34433.03$ грн.;

II. $С_{М2} = С_{М\text{-}Г} \cdot Т_2 = 38.32 \cdot 851.04 = 32611.85$ грн.;

Накладні витрати складають 67% від заробітної плати:

I. $С_{Н1} = 83436{,}68 \cdot 0{,}67 = 55902.57$ грн.;

II. $С_{Н2} = 79024.17 \cdot 0,67 = 52946.19$ грн.;

Отже, вартість розробки програмного продукту за варіантами становить:

I. $С_{ПП1} = 83436,68 + 18356.06 + 34433.03 + 55902.57 = 192128.34$ грн.;

II. $С_{ПП2} = 79024.17 + 17385.31 + 32611.85 + 52946.19 = 181967.52$ грн.;

4.7 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$К_{ТЕР1} = 2,185 / 192128.34 = 1,137 \cdot 10^{-5}$;

$К_{ТЕР2} = 1,8558 / 181967.52 = 1,02 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР2} = 1,137 \cdot 10^{-5}$.

4.8 Висновки до розділу 5

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їхважливості.

Було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що

залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1,137 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

− мова програмування –Java;

− СКБДPostgreSQL;

− інтерфейс користувача, створений за технологією Spring.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, функціонал і швидкодію.

ВИСНОВКИ

В роботі було досліджено сучасний стан сфери надання спортивних послуг, та автоматизацій адміністрування. Була проведена пошукова робота та опрацьовані існуючі дослідження та напрацювання. Був зроблений аналіз рику сфері надання послуг, на в сфері програмного забезпечення для автоматизації.

Були сформульовані головні завдання дипломної роботи та загальна її структура. Згідно до основних задач дипломної роботи були сформульовані вимоги до програмного продукту що має бути розробленим в рамках дипломної роботи.

В процесі проходження практики, були поглиблені теоретичні знання з автоматизації, методів, які використовуються для адміністрування, для веб програмування. Був проведений огляд найпопулярніших бібліотек, та програмних засобів, які дозволяли вирішувати деякі проблеми розглянуті в проблематиці.

Були вивчені нові технології та підготовлена програмна платформа для автоматизації адміністрування та розвитку сфери спорту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Азоева В. Я. Управление организацией. – Москва: ИНФРА-М, 1998. – 669 с.
2. Алгазинов Е.К. Анализ и компьютерное моделирование информационных процессов и систем / Е.К. Алгазинов, А.А. Сирота. – Москва: Диалог МИФИ, 2009. – 414 с.
3. Бабанский А.В. Система непрерывного улучшения продуктов и процессов. - Москва: Экоперспектива, 199. – 237 с.
4. Басовский Л. Е., Протасьев В.В. Управление качеством: Учебник. - М.: Инфра-М, 2000. – 212с.
5. Бєлінський П.Ю. Менеджмент виробництва та операцій: Навчальний посібник - Чернівці: Рута, 2000. – 220 с.
6. Блинов А.М. Информационная безопасность / А.М. Блинов. – СПб.: Изд-во СПбГУЭФ, 2010. – 96 с.
7. Василенко АО., Ткаченко TL Виробничий (операційний) менеджмент: Навчальний посібник. - Київ: ЦУЛ, 2003. – 532 с.
8. Верба В.А., Загородыіх О.А. Проектний аналіз: Підручник. - К.-КНЕУ, 2000. – 322 с.
9. Гевко і.Б:. Операційний менеджмент: Навчальний посібник: - К.: Кондор, 2005. – 228 с.
10. Горнаков С. Г. Осваиваем популярные системы управления сайтом / Горнаков С. Г. – ДМК Пресс, 2009. – 113 с.
11. Кобиляцький Л.С Управління проектами: Навч. Пос1б.-К. :МАУП, 2002. – 200 с.

ДОДАТОК А ДЕМОНСТРАЦІЙНІ МАТЕРІАИ

Система автоматизації адміністрування сайту

Виконав: Жуковський Андрій Віталійович
Науковий керівник: асистет кафедри ММСА
Древаль Максим Михайлович

Постановка задачі

- Виконати дослідження ринку в сфері надання спортивних послуг, ознайомитися з потрібностями в адмініструванні в даній сфері

- Проаналізувати методи та програмне забезпечення яке може використовуватись в автоматизації адміністрування

- Застосувати вибрані методи,та розробити програмний продукт

Актуальність роботи

Актуалність теми можна підкреслити тим, що сфера спорту зажди була більш практична ніж теоритична, чи технічна. Саме через це технології які задіяні в більшості сфер життя людини, нажаль відсутні в сфері спорту.

Розробка спеціального програмного засоба для автоматизування адміністрвання домоможе не тільки зменшити витрати, та й загалом поліпшить рівень сфери надання послуг.

Об’єкт дослідження

Інсуючі засоби для автоматизації адміністрування в сфері надання спортивних послуг

Предмет дослідження

Предмет дослідження - методи автоматизації та адімінстурвання, існуючи програмні засоби для адміністративних робіт.

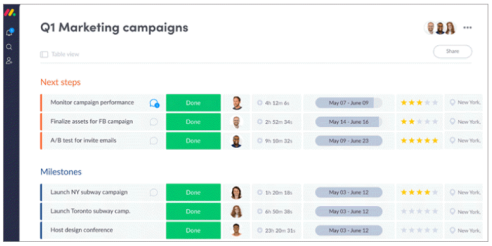
Мета роботи

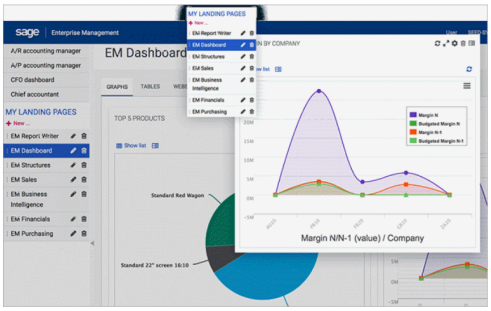
Розробка програмного продукту з необхідним функціоналом для вирішення проблематики дипломної роботи

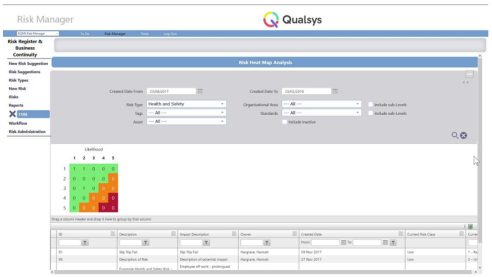
Адміністрування в сфера спорту

У сучасній науці під “управлінням” розуміється процес керівництва або управління працівником, робочою групою, колективом або різними організаціями, які працюють в умовах ринкової економіки. Управління спортом - це самостійний вид професійної діяльності, який спрямований на досягнення цілей спортивної організації, яка працює в ринковому середовищі за рахунок раціонального використання матеріальних, трудових та інформаційних ресурсів.

Існуючи засоби для менаджмента





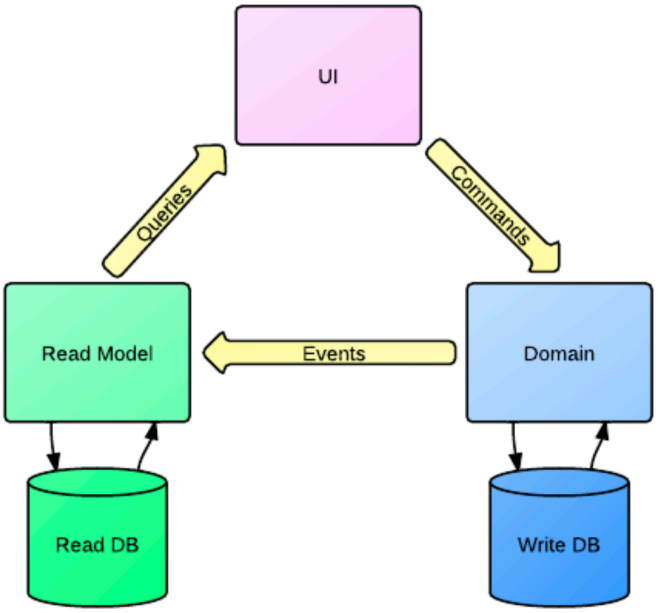


Архітектурний стиль REST

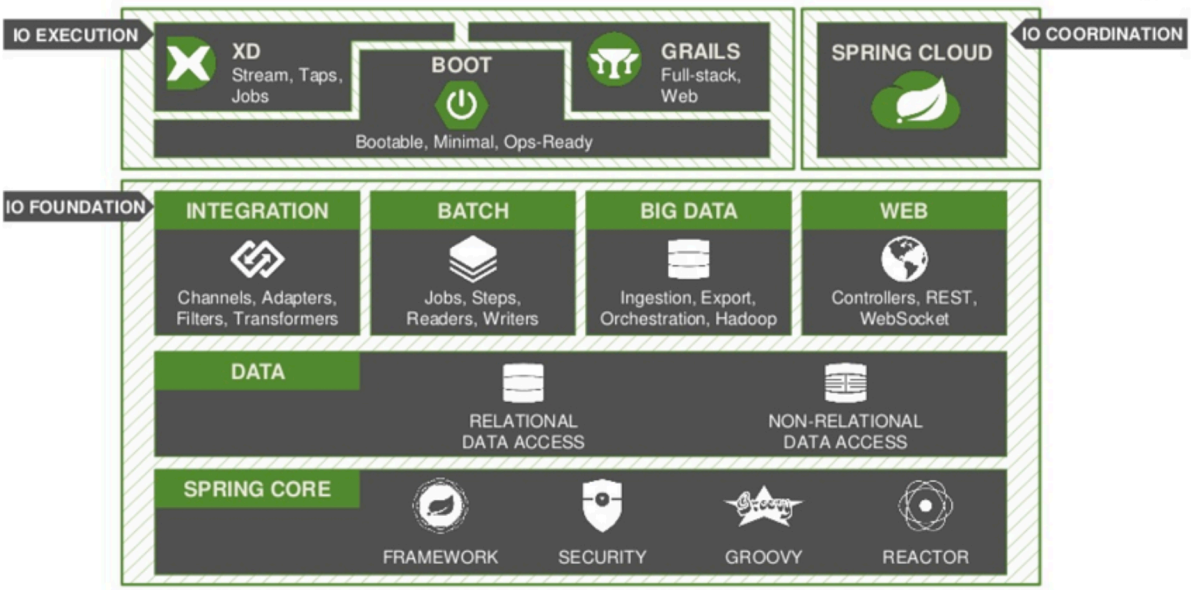
Абревіатура REST розшифровується як representational state transfer - «передача стану уявлення» або, краще сказати, представлення даних в зручному для клієнта форматі. Термін "REST" був введений Роєм Філдінгом в 2000 р Основна ідея REST в тому, що кожне звернення до сервісу переводить клієнтську програму в новий стан.

Характеристика REST

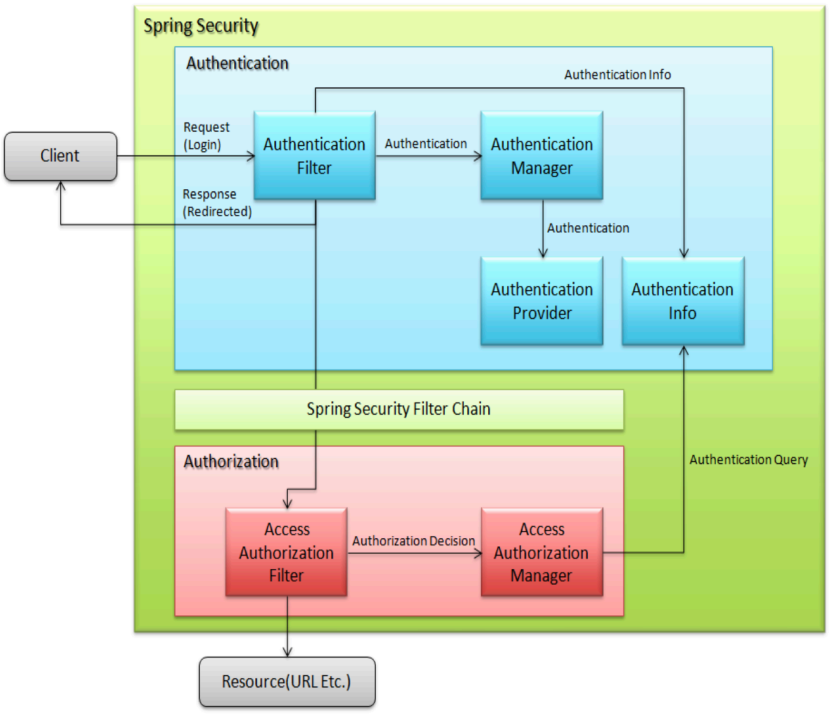
- Технологія створення веб-сервісів і web API
- Цільова аудиторія: публічний доступ, на противагу SOAP / WS
- Statelessness, відсутність зберігання стану і сесій
- Багаторівнева архітектура з кешування з коробки (GET запити кешуються)
- Взаємодія з клієнтом через запит-відповідь, коди відповіді можуть сказати як пройшов запит
- Безпека через HTTPS і OAuth / OpenID
- Широкий спектр клієнтів на різних платформах і технологіях. За рахунок HTTP відмінно використовується з мобільних пристроїв.


UI
Queries
Commands
Read Model
Events
Domain
Read DB
Write DB

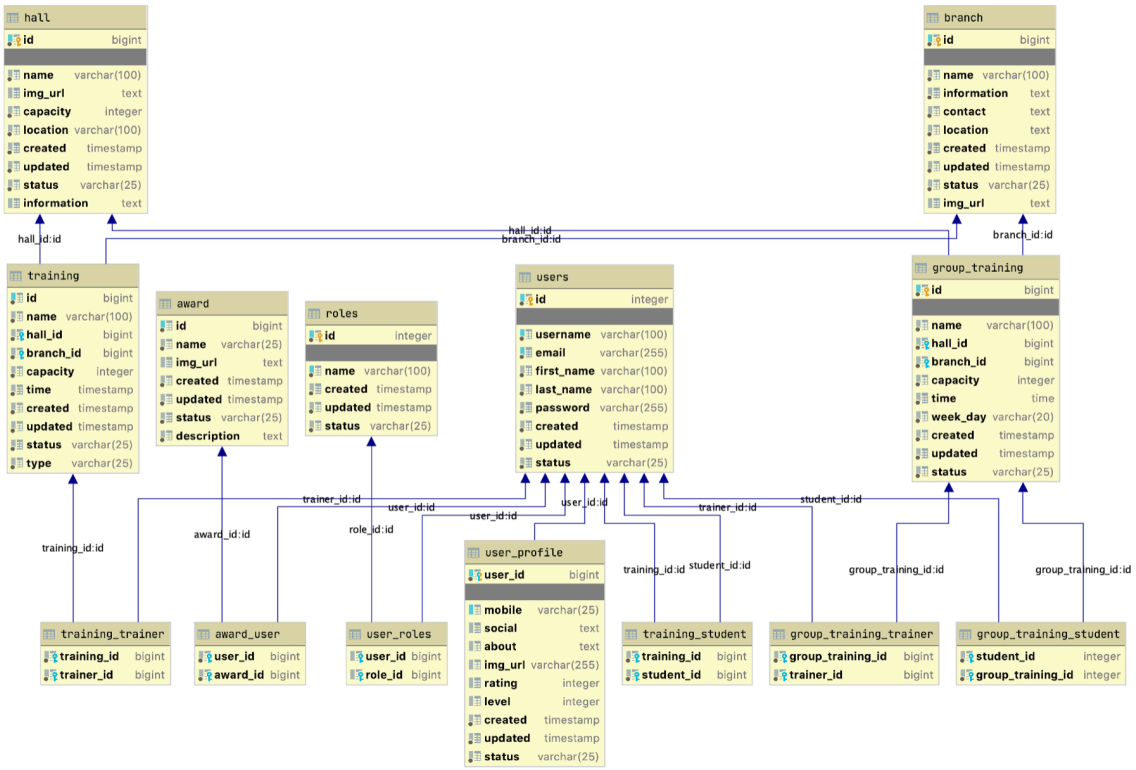
Spring IO Platform



Аутентифікація



Стурктура бази даних



Висновки

• Було проведено аналіз ринку надання спортивих послуг, методи які використовуються для адміністрування в данній сфері. Був проведенний аналіз програмного забезпеченн, який використовуються в даній сфері, та сближенних до неї

• В результаті було виявленно необхідних функціона та методи для поліпшення й автоматизування адміністування в сфері надання спротивних послуг

• Створено веб-сервіс для автоматизації адміністрування та загаом для адміністративних задач, для танцювальної організації

Шляхи подальшого розвитку

У ході дослідження та реалзіації програмного продкуту було прийнято рішеня розробити веб-сервіс, який є легко доповнюванним, та еластичним до реалізації нових задач та функцій.
В подальших роботах планується перехід на мобільні платформи, та інтеграція з соціальними мережами, таких як Telegram, Viber.

Дякую за увагу!

ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthentica-
tionToken;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping(value = "/api/v1/auth/")
public class AuthenticationRestControllerV1 {
    private final AuthenticationManager authenticationManager;
    private final JwtTokenProvider jwtTokenProvider;
    private final UserService userService;
    private final UserProfileService userProfileService;

    @Autowired
    public AuthenticationRestControllerV1(AuthenticationManager authentication-
Manager, JwtTokenProvider jwtTokenProvider, UserService userService, UserPro-
fileService userProfileService) {
        this.authenticationManager = authenticationManager;
        this.jwtTokenProvider = jwtTokenProvider;
        this.userService = userService;
        this.userProfileService = userProfileService;
```

```
    }

    @PostMapping("login")
    public ResponseEntity login(@RequestBody AuthenticationRequestDto requestDto) {
        try {
            String username = requestDto.getUsername();
            User user = userService.findByUsername(username);
            System.out.println(user == null);

            if (user == null) {
                System.out.println("return");
                //return (ResponseEntity) ResponseEntity.badRequest();
               //return ResponseEntity.status(406).build();
                throw new BadCredentialsException("Invalid username or password");
               //throw new BadCredentialsException
               //UsernameNotFoundException("User with username: " + username + " not found");
            }

                System.out.println("try");

            authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, requestDto.getPassword()));

            String token = jwtTokenProvider.createToken(username, user.getRoles());

            Map<Object, Object> response = new HashMap<>();
            response.put("username", username);
            response.put("user_id",user.getId());
```

```
            response.put("role",jwtTokenProvider.getRoleNames(user.getRoles()));
            response.put("token", token);

            return ResponseEntity.ok(response);
        } catch (AuthenticationException e) {
            // throw  new UsernameNotFoundException("User  not found");
             //
            throw new BadCredentialsException("Invalid username or password");
        }
    }

    @PostMapping("register")
    public ResponseEntity register(@RequestBody RegisterUserDto userDto) {
        try {
            System.out.println(userDto.toString());
            String username = userDto.getUsername();
            User checkUser = userService.findByUsername(userDto.getUsername());

            if(checkUser!=null){
                throw new UsernameNotFoundException("Username: " + username + " is not available");
            }
            User user = userService.register(userDto.toUser());

            if (user == null) {
                throw new UsernameNotFoundException("User with username: " + username + " can't register");
            }

            UserProfile userProfile = new UserProfile();
```

```
            userProfile.setUserId(user.getId());
            userProfileService.register(userProfile);

            authenticationManager.authenticate(new UsernamePasswordAuthentication-
Token(username, userDto.getPassword()));
            String token = jwtTokenProvider.createToken(username, user.getRoles());
            Map<Object, Object> response = new HashMap<>();
            response.put("username", username);
            response.put("user_id",user.getId());
            response.put("email",user.getEmail());
            response.put("firstName",user.getFirstName());
            response.put("lastName",user.getLastName());
            response.put("role",jwtTokenProvider.getRoleNames(user.getRoles()));
            response.put("token", token);
            return ResponseEntity.ok(response);
        } catch (AuthenticationException e) {
            //throw  new UsernameNotFoundException("User  not found");
            throw new BadCredentialsException("Invalid username or password");
        }
    }

    @PostMapping("checkUsername")
    public ResponseEntity checkUsername(@RequestBody CheckUsernameDto
checkUsernameDto){
        Map<Object, Object> response = new HashMap<>();
        response.put("isExist",userService.checkUsername(checkUsernameDto.get-
Username()));
        return ResponseEntity.ok(response);
    }
```

```
    @PostMapping("checkEmail")
    public ResponseEntity checkEmail(@RequestBody CheckEmailDto checkEmailDto){
        Map<Object, Object> response = new HashMap<>();
        response.put("isExist",userService.checkEmail(checkEmailDto.getEmail()));
        return ResponseEntity.ok(response);
    }

    @PostMapping("checkMobile")
    public ResponseEntity checkMobile(@RequestBody CheckMobileDto checkMobileDto){
        Map<Object, Object> response = new HashMap<>();
        response.put("isExist",userProfileService.checkMobile(checkMobileDto.getMobile()));
        return ResponseEntity.ok(response);
    }
}

import com.zuk.dto.training.GroupTrainingDto;
import com.zuk.model.WeekDay;
import com.zuk.service.GroupTrainingService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping(value = "/api/v1/admin/training/groupTraining/")
public class AdminGroupTrainingControllerV1 {
```

```
    private final GroupTrainingService groupTrainingService;

    @Autowired
    public AdminGroupTrainingControllerV1(GroupTrainingService groupTraining-
Service) {
        this.groupTrainingService = groupTrainingService;
    }

    @GetMapping(value = "{id}")
    public ResponseEntity findById(@PathVariable(name = "id") Long id){

        return new ResponseEntity<>(GroupTrainingDto.fromGroupTrainingWithTrain-
er (groupTrainingService.findById(id)), HttpStatus.OK);
    }

    @GetMapping(value = "")
    public ResponseEntity findAll(){

        return new ResponseEntity<>(GroupTrainingDto.fromArrayGroupTrainingAd-
min(groupTrainingService.findAll()), HttpStatus.OK);
    }

    @GetMapping(value = "findByDay")
    public ResponseEntity findByDay(@RequestParam WeekDay weekDay){

        return new ResponseEntity<>(GroupTrainingDto.fromArrayGroupTrainingAd-
min(groupTrainingService.findByDay(weekDay)), HttpStatus.OK);
    }
    @GetMapping(value = "findByBranch")
    public ResponseEntity findByBranch(@RequestParam Long branchId){
```

```
        return new ResponseEntity<>(GroupTrainingDto.fromArrayGroupTrainingAd-
min(groupTrainingService.findByBranch(branchId)), HttpStatus.OK);
    }

    @PostMapping(value = "create")
    public ResponseEntity create(@RequestBody GroupTrainingDto groupTraining-
Dto){
        return new ResponseEntity<>(groupTrainingService.create(groupTrainingDto.-
toGroupTrainingAdmin()),HttpStatus.OK);
    }

    @PostMapping(value = "update")
    public ResponseEntity update(@RequestBody GroupTrainingDto groupTraining-
Dto){
        return new
ResponseEntity<>(GroupTrainingDto.fromGroupTrainingAdmin(groupTrainingSer-
vice.update(groupTrainingDto.toGroupTrainingAdminWithId())),HttpStatus.OK);
    }

    @PostMapping(value = "setTrainer")
    public ResponseEntity setTrainer(@RequestBody GroupTrainingDto groupTrain-
ingDto){
        return new ResponseEntity<>(GroupTrainingDto.fromGroupTrainingWithTrain-
er(groupTrainingService.saveWithGroupTrainingTrainer(groupTrainingDto.getId(),gr
oupTrainingDto.getTrainerIds())),HttpStatus.OK);
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping(value = "/api/v1/admin/training/groupTraining/")
public class AdminGroupTrainingControllerV1 {

    private final GroupTrainingService groupTrainingService;

    @Autowired
    public AdminGroupTrainingControllerV1(GroupTrainingService groupTraining-
Service) {
        this.groupTrainingService = groupTrainingService;
    }

    @GetMapping(value = "{id}")
    public ResponseEntity findById(@PathVariable(name = "id") Long id){

        return new ResponseEntity<>(GroupTrainingDto.fromGroupTrainingWithTrain-
er (groupTrainingService.findById(id)), HttpStatus.OK);
    }

    @GetMapping(value = "")
    public ResponseEntity findAll(){

        return new ResponseEntity<>(GroupTrainingDto.fromArrayGroupTrainingAd-
min(groupTrainingService.findAll()), HttpStatus.OK);
    }

    @GetMapping(value = "findByDay")
```

```
    public ResponseEntity findByDay(@RequestParam WeekDay weekDay){

        return new ResponseEntity<>(GroupTrainingDto.fromArrayGroupTrainingAd-
min(groupTrainingService.findByDay(weekDay)), HttpStatus.OK);
    }
    @GetMapping(value = "findByBranch")
    public ResponseEntity findByBranch(@RequestParam Long branchId){
        return new ResponseEntity<>(GroupTrainingDto.fromArrayGroupTrainingAd-
min(groupTrainingService.findByBranch(branchId)), HttpStatus.OK);
    }

    @PostMapping(value = "create")
    public ResponseEntity create(@RequestBody GroupTrainingDto groupTraining-
Dto){
        return new ResponseEntity<>(groupTrainingService.create(groupTrainingDto.-
toGroupTrainingAdmin()),HttpStatus.OK);
    }

    @PostMapping(value = "update")
    public ResponseEntity update(@RequestBody GroupTrainingDto groupTraining-
Dto){
        return new
ResponseEntity<>(GroupTrainingDto.fromGroupTrainingAdmin(groupTrainingSer-
vice.update(groupTrainingDto.toGroupTrainingAdminWithId())),HttpStatus.OK);
    }

    @PostMapping(value = "setTrainer")
    public ResponseEntity setTrainer(@RequestBody GroupTrainingDto groupTrain-
ingDto){
```

```
        return new ResponseEntity<>(GroupTrainingDto.fromGroupTrainingWithTrainer(groupTrainingService.saveWithGroupTrainingTrainer(groupTrainingDto.getId(),groupTrainingDto.getTrainerIds())),HttpStatus.OK);
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.core.AuthenticationException;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping(value = "/api/v1/admin/bid/")
public class AdminBidControllerV1 {

    private final FeedBackService feedBackService;

    @Autowired
    public AdminBidControllerV1(FeedBackService feedBackService) {
        this.feedBackService = feedBackService;
    }

    @GetMapping(value = "findAllFeedback/")
```

```
public ResponseEntity findAllFeedback()
{
   Map<Object, Object> response = new HashMap<>();
   try {

      ArrayList result = feedBackService.findAll();
      if (result.size() == 0) {
         response.put("error","No feedback bid");
         return new ResponseEntity<>(response,HttpStatus.OK);
      }



      return new ResponseEntity<>(result, HttpStatus.OK);
   } catch (AuthenticationException e) {
      throw new BadCredentialsException("Invalid username or password");
   }
}

@GetMapping(value = "findAllNotActiveFeedback/")
public ResponseEntity findAllNotActiveFeedback()
{
   Map<Object, Object> response = new HashMap<>();
   try {

      ArrayList result = feedBackService.findAllNotActive();
      if (result.size() == 0) {
         response.put("error","No Not_Active feedback bid");
         return new ResponseEntity<>(response,HttpStatus.OK);
      }
```

```
            return new ResponseEntity<>(result, HttpStatus.OK);
        } catch (AuthenticationException e) {
            throw new BadCredentialsException("Invalid username or password");
        }
    }

    @PostMapping(value = "applyFeedback/")
    public ResponseEntity setActive(@RequestBody FeedBackDto feedBackDto)
    {
        Map<Object, Object> response = new HashMap<>();
        try {

            Feedback feedback =
feedBackService.adminApplyById(feedBackDto.getId());
            if (feedback == null) {
                response.put("error","Can`t apply feedback");
                return new ResponseEntity<>(response,HttpStatus.OK);
            }



            return new ResponseEntity<>(feedback, HttpStatus.OK);
        } catch (AuthenticationException e) {
            throw new BadCredentialsException("Invalid username or password");
        }
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;



@RestController
@RequestMapping(value = "/api/v1/admin/branch/")
public class AdminBranchControllerV1 {
    private final BranchService branchService;

    @Autowired
    public AdminBranchControllerV1(BranchService branchService) {
        this.branchService = branchService;
    }



    @PostMapping(value = "update")
    public ResponseEntity updateById(@RequestBody BranchDto branchDto){

        Branch branch = branchDto.toBrunch();
        branch.setId(branchDto.getId());
        branch.setStatus(branchDto.getStatus());
        return new ResponseEntity<>(branchService.update(branch), HttpStatus.OK);
    }

    @PostMapping(value = "create")
    public ResponseEntity create(@RequestBody BranchDto branchDto){
        return new ResponseEntity<>(branchService.create(branchDto.toBrunch()),
HttpStatus.OK);
```

```
    }


    @GetMapping(value = "")
    public ResponseEntity<List<BranchDto>> findAll(){

        return new ResponseEntity<List<BranchDto>>(BranchDto.fromArrayBranch-
Admin(branchService.findAll()), HttpStatus.OK);
    }

    @GetMapping(value = "{id}" )
    public ResponseEntity findById(@PathVariable(name = "id") long id){
        System.out.println("good");
        return new ResponseEntity<>(branchService.findById((long) id), HttpSta-
tus.OK);
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping(value = "/api/v1/admin/hall/")
public class AdminHallControllerV1 {

    private final HallService hallService;
```

```
    @Autowired
    public AdminHallControllerV1(HallService hallService) {
        this.hallService = hallService;
    }

    @GetMapping(value = "{id}")
    public ResponseEntity findById(@PathVariable(name = "id") Long id){

        return new ResponseEntity<>(HallDto.fromHallToAdminDto(hallService.find-
ById(id)), HttpStatus.OK);
    }

    @GetMapping(value = "")
    public ResponseEntity findAll(){

        return new ResponseEntity<>(HallDto.fromArrayHallToAdminDto(hallSer-
vice.findAll()), HttpStatus.OK);

    }

    @PostMapping(value = "create/")
    public ResponseEntity create(@RequestBody HallDto hallDto){

        return new ResponseEntity<>(hallService.create(hallDto.toHall()), HttpSta-
tus.OK);
    }

    @PostMapping(value = "update/")
    public ResponseEntity update(@RequestBody HallDto hallDto){
```

```
        return new ResponseEntity<>(hallService.update(hallDto.toHallWithId()), Http-
Status.OK);
    }

}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/api/v1/admin/news/")
public class AdminNewsControllerV1 {
    private final NewsService newsService;


    @Autowired
    public AdminNewsControllerV1(NewsService newsService) {
        this.newsService = newsService;
    }

    @GetMapping(value = "{id}")
    public ResponseEntity findById(@PathVariable(name = "id") Long id){

        return new ResponseEntity<>(NewsDto.fromAdminNews(newsService.find-
ById(id)), HttpStatus.OK);
```

```
    }

    @GetMapping(value = "")
    public ResponseEntity findAll(){

        return new ResponseEntity<>(NewsDto.fromAdminArrayNews(newsSer-
vice.findAll()), HttpStatus.OK);
    }

    @PostMapping(value = "create/")
    public ResponseEntity create(@RequestBody NewsDto newsDto){
        return new
ResponseEntity<>(NewsDto.fromAdminNews(newsService.create(newsDto.to-
NewsForCreate())), HttpStatus.OK);
    }

    @PostMapping(value = "update/")
    public ResponseEntity update(@RequestBody NewsDto newsDto){
        return new
ResponseEntity<>(NewsDto.fromAdminNews(newsService.updated(newsDto.to-
NewsWithStatus())), HttpStatus.OK);
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.core.AuthenticationException;
import org.springframework.web.bind.annotation.*;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping(value = "/api/v1/admin/trainer")
public class AdminTrainerControllerV1 {

    private final UserService userService;
    private final JwtTokenProvider jwtTokenProvider;

    @Autowired
    public AdminTrainerControllerV1(UserService userService, JwtTokenProvider
jwtTokenProvider) {
        this.userService = userService;
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @GetMapping(value = "findByFirstName/")
    public ResponseEntity getTrainerByFirstName(@RequestParam String firstName )
    {
        try {

            Map<Object, Object> response = new HashMap<>();
            ArrayList result = userService.findTrainerByFirstName(firstName);
            if (result.size() == 0) {
                response.put("error","Nothing to show");
                return new ResponseEntity<>(response,HttpStatus.OK);
```

```
        }


        return new ResponseEntity<>(AdminUserWithRoleDto.fromArrayUser(result), HttpStatus.OK);
      } catch (AuthenticationException e) {
        throw new BadCredentialsException("Invalid username or password");
      }
    }
    @GetMapping(value = "findByLastName/")
    public ResponseEntity getUserByLastName(@RequestParam String lastName)
    {
      try {

        Map<Object, Object> response = new HashMap<>();

        ArrayList result = userService.findTrainerByLastName(lastName);
        if (result.size() == 0) {
          response.put("error","Nothing to show");
          return new ResponseEntity<>(response,HttpStatus.OK);
        }

        return new ResponseEntity<>(AdminUserWithRoleDto.fromArrayUser(result), HttpStatus.OK);
      } catch (AuthenticationException e) {
        throw new BadCredentialsException("Invalid username or password");
      }
    }
    @GetMapping(value = "findByFirstNameAndLastName/")
```

```
    public ResponseEntity getUserByFirstNameAndLastName(@RequestParam String firstName, String lastName)
    {
        try {
            Map<Object, Object> response = new HashMap<>();

            ArrayList result = userService.findTrainerByFirstNameAndLastName(firstName,lastName);
            if (result.size() == 0) {
                response.put("error","Nothing to show");
                return new ResponseEntity<>(response,HttpStatus.OK);
            }

            return new ResponseEntity<>(AdminUserWithRoleDto.fromArrayUser(result), HttpStatus.OK);
        } catch (AuthenticationException e) {
            throw new BadCredentialsException("Invalid username or password");
        }
    }
    @GetMapping(value = "findByUsername/")
    public ResponseEntity getTrainerByUsername(@RequestParam String username)
    {
        User user = userService.findTrainerByUsername(username);
        if(user==null){
            Map<Object, Object> response = new HashMap<>();
            response.put("error","Nothing to show");
            return new ResponseEntity<>(response,HttpStatus.OK);
        }
        return new ResponseEntity<>(AdminUserWithRoleDto.fromUser(user), HttpStatus.OK);
```

```
}


@GetMapping(value = "{id}")
public ResponseEntity getTrainerById(@PathVariable(name = "id") Long id) {
    try {
        User user = userService.findTrainerById(id);

        Map<Object, Object> response = new HashMap<>();

        if (user == null) {
            response.put("error","Nothing to show");
            return new ResponseEntity<>(response,HttpStatus.OK);
        }

        AdminUserWithRoleDto result = AdminUserWithRoleDto.fromUser(user);

        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (AuthenticationException e) {
        //throw  new UsernameNotFoundException("User  not found");
        throw new BadCredentialsException("Invalid username or password");
    }
}

@GetMapping(value = "")
public ResponseEntity getUser() {
    try {

        List<User> userList = userService.findAllTrainer();
```

```
        Map<Object, Object> response = new HashMap<>();
        if (userList.size() == 0) {
            response.put("error","Nothing to show");
            return new ResponseEntity<>(response,HttpStatus.OK);
        }

        ArrayList<AdminUserWithRoleDto> result = AdminUserWithRoleDto.from-
ArrayUser(userList);

        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (AuthenticationException e) {
        throw new BadCredentialsException("Invalid username or password");
    }
}

@GetMapping(value = "activateUser/")
public ResponseEntity activateUser(@RequestBody UserDto userDto) {
    try {
        return new ResponseEntity<>(AdminUserWithRoleDto.fromUser(userSer-
vice.activateUser(userDto.getId())), HttpStatus.OK);
    } catch (AuthenticationException e) {
        throw new BadCredentialsException("Invalid username or password");
    }

}

@GetMapping(value = "setRoleTrainer")
public ResponseEntity setRoleTrainer(@RequestParam long id){

    User user = userService.setRoleTrainer(id);
```

```
        return new ResponseEntity<>(UserDto.fromUserAdmin(user,jwtToken-
Provider.getRoleNames(user.getRoles())), HttpStatus.OK);
    }
}

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Component
@EnableWebMvc
public class CORsFilter implements Filter, WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**");
    }

    @Override
```

```
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
throws IOException, ServletException {

        HttpServletResponse response = (HttpServletResponse) res;
        HttpServletRequest request = (HttpServletRequest) req;
        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OP-
TIONS, DELETE");
        response.setHeader("Access-Control-Allow-Headers", "Content-Type, Access-
Control-Allow-Headers, Authorization, X-Requested-With,observe");
        response.setHeader("Access-Control-Max-Age", "3600");
        response.setHeader("Access-Control-Allow-Credentials", "true");
        response.setHeader("Access-Control-Expose-Headers", "Authorization");
        response.addHeader("Access-Control-Expose-Headers", "responseType");
        response.addHeader("Access-Control-Expose-Headers", "observe");
        chain.doFilter(req, res);
    }

    public void init(FilterConfig filterConfig) {}

    public void destroy() {}

}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
```

```
import org.springframework.security.config.annotation.web.configuration.WebSecu-
rityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private final JwtTokenProvider jwtTokenProvider;

    private static final String ADMIN_ENDPOINT = "/api/v1/admin/**";
    private static final String AUTH_ENDPOINT = "/api/v1/auth/**";
    private static final String BRANCH_ENDPOINT = "/api/v1/branch/**";
    private static final String TRAINING_ENDPOINT = "/api/v1/training/**";
    private static final String TRAINERPUBLIC_ENDPOINT = "/api/v1/userPublic/
**";
    private static final String HALLPUBLIC_ENDPOINT = "/api/v1/hallPublic/**";
    private static final String NEWS_ENDPOINT = "/api/v1/news/**";
    private static final String USERS_ENDPOINT = "/api/v1/users/**";
    private static final String TOKEN_ENDPOINT = "/api/v1/token/**";
    private static final String FEEDBACK_ENDPOINT = "/api/v1/feedback/**";

    @Autowired
    public SecurityConfig(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
```

```
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .httpBasic().disable()
                .cors().and()
                .csrf().disable()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.S-
TATELESS)
                .and()
                .authorizeRequests()
                .antMatchers(AUTH_ENDPOINT).permitAll()
                .antMatchers(FEEDBACK_ENDPOINT).permitAll()
                .antMatchers(TOKEN_ENDPOINT).permitAll()
                .antMatchers(BRANCH_ENDPOINT).permitAll()
                .antMatchers(TRAINING_ENDPOINT).permitAll()
                .antMatchers(TRAINERPUBLIC_ENDPOINT).permitAll()
                .antMatchers(HALLPUBLIC_ENDPOINT).permitAll()
                .antMatchers(NEWS_ENDPOINT).permitAll()
                .antMatchers(USERS_ENDPOINT).hasRole("USER")
                .antMatchers(ADMIN_ENDPOINT).hasRole("ADMIN")
                .anyRequest().authenticated()
                .and()
                .apply(new JwtConfigurer(jwtTokenProvider));
    }
}

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.zuk.dto.admin.AdminUserWithRoleDto;
```

```
import com.zuk.model.*;
import lombok.Data;
import java.sql.Time;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

@Data
@JsonIgnoreProperties(ignoreUnknown = true)
public class GroupTrainingDto {
    private Long id;
    private String name;
    private Long branchId;
    private int capacity;
    //private String location;
    private Long hallId;
    private Time time;
    private WeekDay weekDay;
    private Timestamp created;
    private Timestamp updated;
    private Status status;
    private List<Long> trainerIds;


    public static GroupTrainingDto fromGroupTrainingAdmin(GroupTraining group-
Training){
        GroupTrainingDto groupTrainingDto = new GroupTrainingDto();
        groupTrainingDto.setId(groupTraining.getId());
        groupTrainingDto.setName(groupTraining.getName());
        groupTrainingDto.setBranchId(groupTraining.getBranch().getId());
```

```
        groupTrainingDto.setHallId(groupTraining.getHall().getId());
        groupTrainingDto.setCapacity(groupTraining.getCapacity());
        groupTrainingDto.setTime(groupTraining.getTime());
        groupTrainingDto.setWeekDay(groupTraining.getWeekDay());
        groupTrainingDto.setCreated(groupTraining.getCreated());
        groupTrainingDto.setUpdated(groupTraining.getUpdated());
        groupTrainingDto.setStatus(groupTraining.getStatus());
        return groupTrainingDto;
    }

    public static GroupTrainingDto fromGroupTrainingWithAdminTrainer(Group-
TTraining groupTraining){
        GroupTrainingDto groupTrainingDto = fromGroupTrainingAdmin(groupTrain-
ing);
        List<User> list =  groupTraining.getTrainer();
        List<Long> ids = new ArrayList<>();
        for(User user:list){
            ids.add(user.getId());
        }
        groupTrainingDto.setTrainerIds(ids);
        return groupTrainingDto;
    }

    public static GroupTrainingDto fromGroupTrainingWithTrainer(GroupTraining
groupTraining){
        GroupTrainingDto groupTrainingDto = fromGroupTraining(groupTraining);
        List<User> list =  groupTraining.getTrainer();
        List<Long> ids = new ArrayList<>();
        for(User user:list){
            ids.add(user.getId());
```

```
        }
        groupTrainingDto.setTrainerIds(ids);
        return groupTrainingDto;
    }

    public static GroupTrainingDto fromGroupTraining(GroupTraining groupTraining)
{
        GroupTrainingDto groupTrainingDto = new GroupTrainingDto();
        groupTrainingDto.setId(groupTraining.getId());
        groupTrainingDto.setName(groupTraining.getName());
        groupTrainingDto.setBranchId(groupTraining.getBranch().getId());

        groupTrainingDto.setHallId(groupTraining.getHall().getId());

        groupTrainingDto.setCapacity(groupTraining.getCapacity());
        groupTrainingDto.setTime(groupTraining.getTime());
        groupTrainingDto.setWeekDay(groupTraining.getWeekDay());
        return groupTrainingDto;
    }

    public static ArrayList<GroupTrainingDto>
fromArrayGroupTraining(List<GroupTraining> groupTrainingsList){
        ArrayList<GroupTrainingDto>  groupTrainingDtoArrayList = new
ArrayList<>();
        for (GroupTraining groupTraining:groupTrainingsList){
            groupTrainingDtoArrayList.add(fromGroupTrainingWithTrainer(groupTrain-
ing));
        }
        return groupTrainingDtoArrayList;
    }
```

```
    public static ArrayList<GroupTrainingDto>
fromArrayGroupTrainingAdmin(List<GroupTraining> groupTrainingsList){
        ArrayList<GroupTrainingDto>  groupTrainingDtoArrayList = new
ArrayList<>();
        for (GroupTraining groupTraining:groupTrainingsList){

groupTrainingDtoArrayList.add(fromGroupTrainingWithAdminTrainer(groupTrain-
ing));
        }
        return groupTrainingDtoArrayList;
    }

    public GroupTraining toGroupTrainingAdmin(){
        GroupTraining groupTraining = new GroupTraining();
        //groupTraining.setId(id);
        groupTraining.setName(name);
        Hall hall = new Hall();hall.setId(hallId);
        groupTraining.setHall(hall);
       // groupTraining.setLocation(location);
        Branch branch = new Branch();branch.setId(branchId);
        groupTraining.setBranch(branch);
        groupTraining.setCapacity(capacity);
        groupTraining.setTime(time);
        groupTraining.setWeekDay(weekDay);
        groupTraining.setStatus(status);
        return groupTraining;
    }

    public GroupTraining toGroupTrainingAdminWithId(){
```

```
        GroupTraining groupTraining = new GroupTraining();
        groupTraining.setId(id);
        groupTraining.setName(name);
        Hall hall = new Hall();hall.setId(hallId);
        groupTraining.setHall(hall);
        Branch branch = new Branch();branch.setId(branchId);
        groupTraining.setBranch(branch);
        groupTraining.setCapacity(capacity);
        groupTraining.setTime(time);
        groupTraining.setWeekDay(weekDay);
        groupTraining.setStatus(status);
        return groupTraining;
    }
}

import com.zuk.model.Role;
import com.zuk.model.User;
import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.SQLUpdate;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;



public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String name);
    User findByEmail(String email);
    List<User> findByFirstNameAndLastName(String firstName, String lastName);
    List<User> findByFirstName(String firstName);
    List<User> findByLastName(String lastName);
```

```
    User findByUsernameContains(String name);
    User findByEmailContains(String email);
    List<User> findByFirstNameContains(String firstName);
    List<User> findByLastNameContains(String lastName);

    List<User> findByRolesEquals(Role role);



    User getByIdAndRolesEquals(Long id, Role role);
    User findByUsernameAndRolesEquals(String username, Role role);
    User findByEmailAndRolesEquals(String email, Role role);
    List<User> findByFirstNameAndLastNameAndRolesEquals(String
firstName,String lastName ,Role role);
    List<User> findByFirstNameAndRolesEquals(String firstName , Role role);
    List<User> findByLastNameAndRolesEquals(String lastName, Role role);
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
@Slf4j
public class JwtUserDetailsService implements UserDetailsService {

    private final UserService userService;
```

```
    @Autowired
    public JwtUserDetailsService(@Lazy UserService userService) {
        this.userService = userService;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNot-
FoundException {
        User user = userService.findByUsername(username);

        if (user == null) {
            throw new UsernameNotFoundException("User with username: " + username
+ " not found");
        }

        JwtUser jwtUser = JwtUserFactory.create(user);
        log.info("IN loadUserByUsername - user with username: {} successfully
loaded", username);
        return jwtUser;
    }
}

import com.zuk.model.Role;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jws;
import io.jsonwebtoken.JwtException;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.security.authentication.UsernamePasswordAuthentica-
tionToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import javax.servlet.http.HttpServletRequest;
import java.util.ArrayList;
import java.util.Base64;
import java.util.Date;
import java.util.List;


@Component
public class JwtTokenProvider {

    @Value("${jwt.token.secret}")
    private String secret;
    @Value("${jwt.token.expired}")
    private long validityInMilliseconds;
    @Autowired
    private UserDetailsService userDetailsService;
```

```
    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
        return bCryptPasswordEncoder;
    }
    @PostConstruct
    protected void init() {
        secret = Base64.getEncoder().encodeToString(secret.getBytes());
    }

    public String createToken(String username, List<Role> roles) {
        Claims claims = Jwts.claims().setSubject(username);
        claims.put("roles", getRoleNames(roles));
        Date now = new Date();
        Date validity = new Date(now.getTime() + validityInMilliseconds);
        return Jwts.builder()//
                .setClaims(claims)//
                .setIssuedAt(now)//
                .setExpiration(validity)//
                .signWith(SignatureAlgorithm.HS256, secret)//
                .compact();
    }
    public Authentication getAuthentication(String token) {
        UserDetails userDetails = this.userDetailsService.loadUserByUsername(getUsername(token));
        return new UsernamePasswordAuthenticationToken(userDetails, "", userDetails.getAuthorities());
    }
    public String getUsername(String token) {
```

```
        return
Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody().getSubject();
    }
    public String resolveToken(HttpServletRequest req) {
        String bearerToken = req.getHeader("Authorization");
        if (bearerToken != null && bearerToken.startsWith("Bearer_")) {
            return bearerToken.substring(7, bearerToken.length());
        }
        return null;
    }
    public boolean validateToken(String token) {
        try {
            Jws<Claims> claims =
Jwts.parser().setSigningKey(secret).parseClaimsJws(token);
            if (claims.getBody().getExpiration().before(new Date())) {
                return false;
            }
            return true;
        } catch (JwtException | IllegalArgumentException e) {
            return false;
            //throw new JwtAuthenticationException("JWT token is expired or invalid");
        }
    }

    public List<String> getRoleNames(List<Role> userRoles) {
        List<String> result = new ArrayList<>();

        userRoles.forEach(role -> {
            result.add(role.getName());
        });
```

```
        return result;
    }
}
```